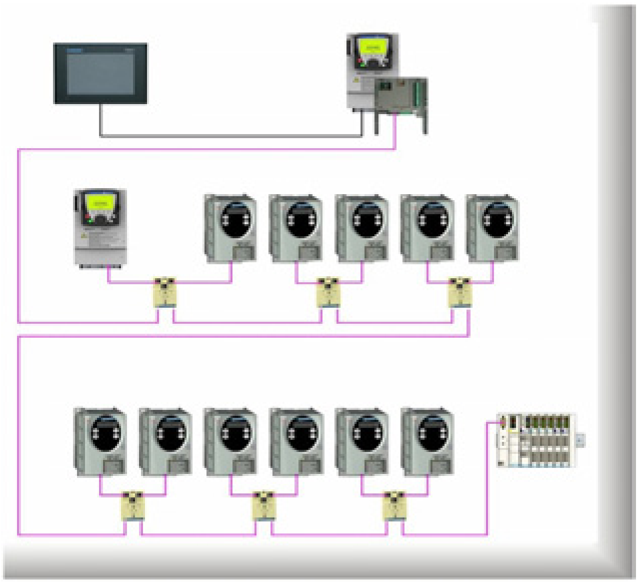
Controller Inside with Altivar Lexium Advantys STB and Magelis

System User Guide

[source code]



33003814.00

Merlin Gerin
Square D
Telemecanique

Schneider Electric
Building a New Electric World

Mar 2006

Contents

Introduction

This document is intended to provide a quick introduction to the described System.
It is **not** intended to replace any specific product documentation. On the contrary, it offers additional information to the product documentation, for installing, configuring and starting up the system.

A detailed functional description or the specification for a specific user application is **not** part of this document. Nevertheless, the document outlines some typical applications where the system might be implemented.

Abbreviations

Word/Expression	Signification
AC	Alternating Current
Advantys	SE product name for a family of I/O modules
Altivar (ATV)	SE product name for a family of VSDs
CANopen	Name for a communications machine bus system
CB	Circuit Breaker
CoDeSys	Hardware-independent IEC 61131-3 programming software
ConneXium	SE product name for a Family of Transparent Factory devices
DC	Direct Current
EDS	Electronic Data Sheet
E-OFF, E-STOP	Emergency Off switch
Harmony	SE product name for a family of switches and indicators
HMI	Human Machine Interface
I/O	Input/Output
IclA (ICLA)	SE product name for a compact drive
Lexium/Lexium05/LXM	SE product name for a family of servo-drives
Magelis	SE product name for a family of HMI-Devices
MB - SL	SE name for a serial Modbus communications protocol
Micro	SE product name for a middle range family of PLCs
NIM	SE product name for a Network Interface Module
PC	Personal Computer
Phaseo	SE product name for a family of power supplies
PLC	Programmable Logic Computer
Powersuite	An SE software product for configuring ALTIVAR drives
Premium	SE product name for a middle range family of PLCs
Preventa	SE product name for a family of safety devices
PS1131 (CoDeSys)	SE Product name for PLC programming software with CoDeSys
PS	Power Supply
SE	Schneider Electric
Sycon	SE product name of a Field bus programming software
Telefast	SE product name for a series of distributed I/O devices
Tesys U	SE product name for a decentralized I/O System
Twido	SE product name of a middle range family of PLCs
TwidoSoft	SE product name for a PLC programming software
Unity (Pro)	SE product name for a PLC programming software
Vijeo Designer	An SE software product for programming Magelis HMI devices
VSD	Variable Speed Drive
WxHxD	Dimensions : Width, Height and Depth
XBT-L1000	An SE software product for programming Magelis HMI devices

Application Source Code

Introduction

Examples of the source code used to attain the system function as described in this document can be downloaded from our „Village“ website under **this** link.

The example source code is in the form of configuration, application and import files. Use the appropriate software tool to either open or import the files

Extension	File Type	Software Tool Required
AIW	Configuration File	Advantys
CNF	**C**onfiguratio**n** **F**ile	Sycon
CO	**C**AN**o**pen definitions file	Sycon
CSV	**C**omma **S**eparated **V**alues, spreadsheet	Twidosoft
CTX		Unity
DCF	**D**evice **C**onfiguration **F**ile	Advantys
DIB	**D**evice **I**ndependent **B**itmap	Sycon
DOC	**Doc**ument file	Microsoft Word
DOP	Project File	Magelis XBTL
EDS	**E**lectronic **D**ata **S**heet – Device Definition	Industrial standard
FEF	Export file	PL7
GSD	EDS file (**G**eraete **S**tamm **D**atei)	Profibus
ISL	**Isl**and file, project file	Advantys
PB	**P**rofi**b**us definitions file	Sycon
PDF	**P**ortable **D**ocument **F**ormat - document	Adobe Acrobat
PRO	Projektdatei	PS1131 - CoDeSys
PS2	Export file	Powersuite export file
RTF	**R**ich **T**ext **F**ile - document	Microsoft Word
STU	Project file	Unity studio
STX	Project file	PL7
TLX	Project file	Twinline control tool
TWD	Project file	TwidoSoft
VDZ	Project file	Vijeo Designer
XEF	Export file	Unity Pro
ZM2	Project File	Zeliosoft

Typical Applications

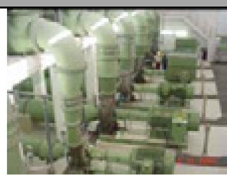
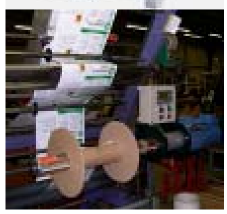
Introduction

Here you will find a list of the typical applications, and their market segments, where this system or subsystem can be applied:

Industry
- Small automated machine or plant components
- Remote automation systems used to supplement large and medium-sized machines

Machines/Services
- Automatic saws
- Automatic winders
- Cartoning machines or carton-folding systems

Foodstuffs/Pharmaceuticals
- Continuous dryers or continuous furnaces

Application	Description	Image
Infrastructure networks	Used in the pumping stations of water supply networks for the purpose of getting water from processing plants to consumers.	
Special-purpose machines	Used cost-effectively on special-purpose machines for assembly, processing, cutting operations, etc. (e.g., winding machines, automated assembly, wood machining).	  
Material conveyors	Used in connection with transportation tasks that involve lifting and shuttling.	

System

Introduction

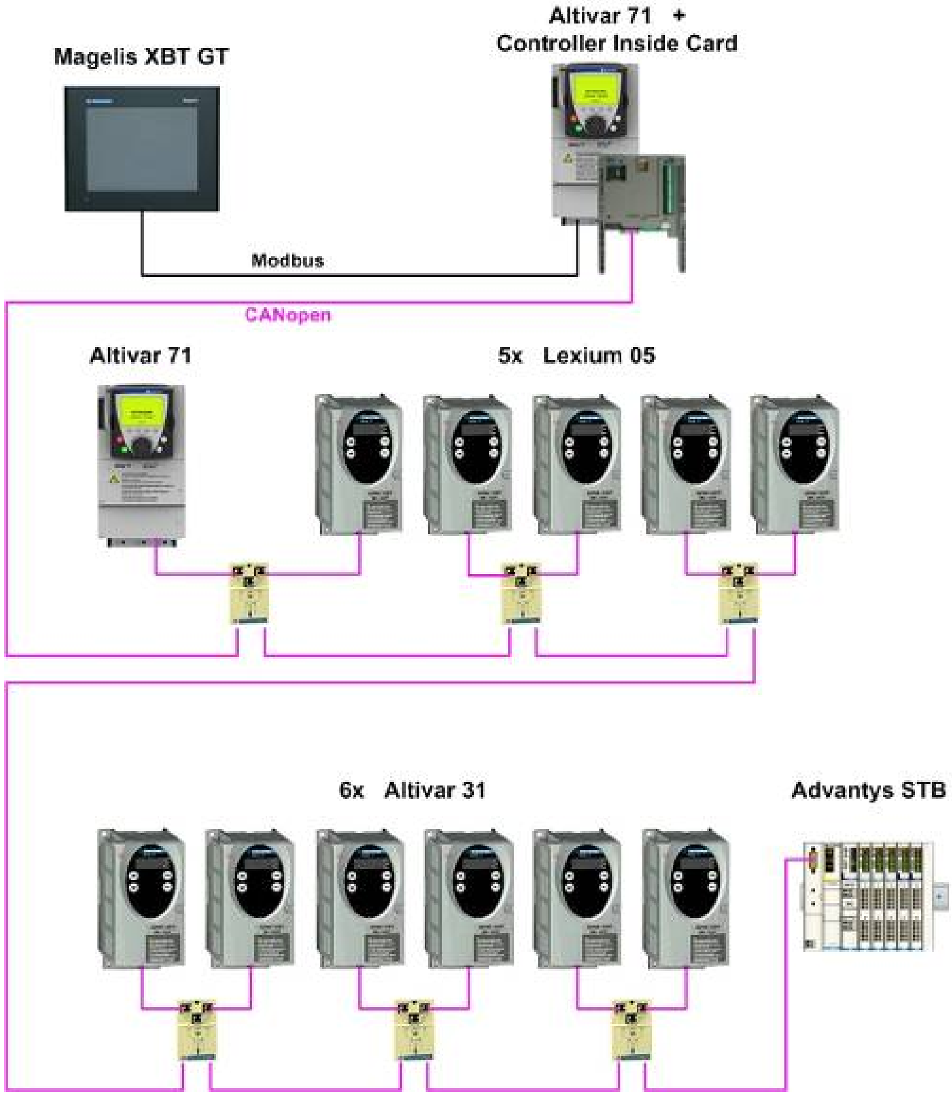
The system chapter describes the architecture, the dimensions, the quantities and different types of components used within this system.

Architecture

General

The control section of this application consists of a Controller Inside PLC installed in an Altivar 71 VSD. Operation at user level is via a connected Magelis HMI panel. The load section is implemented using an Altivar 71 and Altivar 31 variable speed drives, in addition to a Lexium05 servo drive. These are connected to the PLC via the CANopen bus system. The bus system also features an Advantys STB for the connection of various I/O.

Layout



Components

Hardware:
- Master switch (NSC100 Compact)
- 24 V power supply (Phaseo)
- Controller Inside card installed in the Altivar 71
- ATV31 and ATV71 variable speed drives with integrated CANopen interface
- Lexium05 servo drive with integrated CANopen interface
- Motor circuit breaker (GV2)
- Preventa safety relay
- Emergency-off switch (XALK)
- Contactors (LC1D)
- Graphic display terminal (Magelis XBT-GT)
- Advantys STB distributed I/O island
- Standard AC motor
- Servo motor

Software:
- PS1131 (CoDeSys V2.3)
- PowerSuite for ATV31, ATV71 and Lexium05
- Vijeo-Designer V4.30
- Advantys Configuration Tool V2.0

Quantities of Components

For a complete and detailed list of components, the quantities required and the order numbers, please refer to the components list at the rear of this document.

Degree of Protection

Not all the components in this configuration are designed to withstand the same environmental conditions. Some components may need additional protection, in the form of housings, depending on the environment in which you intend to use them. For environmental details of the individual components please refer to the list in the appendix of this document and the appropriate user manual.

Technical-data

Mains voltage	400V AC
Power requirement	~ 15 kW
Drive power rating	2x 0,75 kW, 6x 0,37 kW und 5x 1,4 kW
Motor brake	none
connection	5x 2,5mm² (L1, L2, L3, N, PE)
Safety Level	Cat. 3 (optional)

Safety Notice

The standard and level of safety you apply to your application is determined by your system design and the overall extent to which your system may be a hazard to people and machinery.

As there are no moving mechanical parts in this application example, category 3 (according to EN954-1) has been selected as an optional safety level.

Whether or not the above safety category should be applied to your system should be ascertained with a proper risk analysis.

This document is not comprehensive for any systems using the given architecture and does not absolve users of their duty to uphold the safety requirements with respect to the equipment used in their systems or of compliance with either national or international safety laws and regulations

Dimensions

The dimensions of the devices used (e.g., the PLC (in the VSD), variable speed drive(s), servo drive and power supply) are suitable for installation inside a control cubicle measuring 800x2000x600 mm (WxHxD).
In addition, the display and control elements required to control the system can be integrated into the control cubicle door.

Installation

Introduction

This chapter describes the steps necessary to set up the hardware and configure the software required to fulfill the described function of the application.

Assembly



Note

The application, as configured here, illustrates a possible machine calling for the use of a number of drives with a whole range of different requirements that acquire their input and output data locally via the PLC. Each of the motors is controlled separately by the PLC via the CANopen bus.

The components listed in the next chapter represent a selection of the components required. In particular, the number of motors used and their allocation to variable speed drives and the servo drive are determined by the relevant application (the number of inputs and outputs may also vary).

This document does not, therefore, claim to be comprehensive and does not absolve users of their duty to check the safety requirements of their equipment and to ensure compliance with the relevant national or international rules and regulations in this respect.

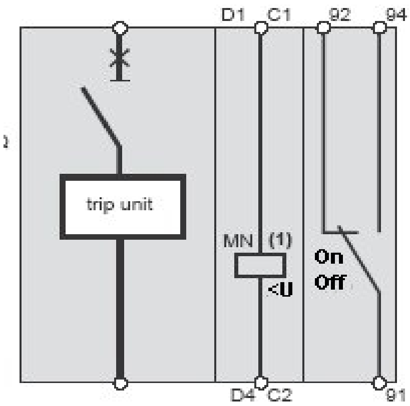
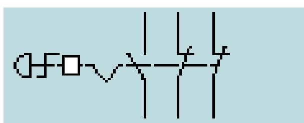
Safety Category 3 is suggested here as one possible option. It is not necessarily binding and not necessarily required for all applications. A proper risk analysis, in accordance with national and/or international standards and regulations, should be produced and verified for each individual system.

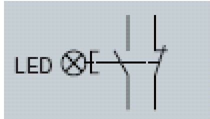
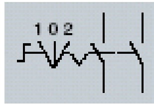
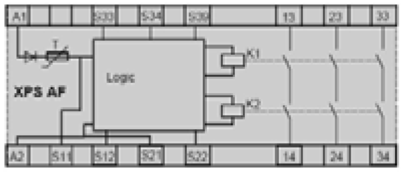
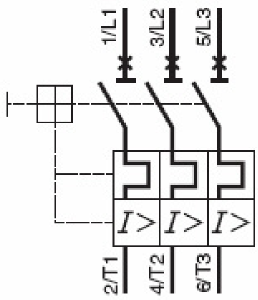
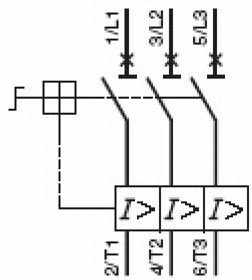
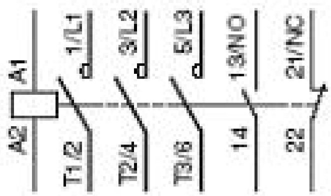
Hardware

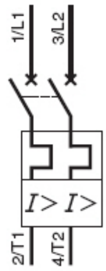
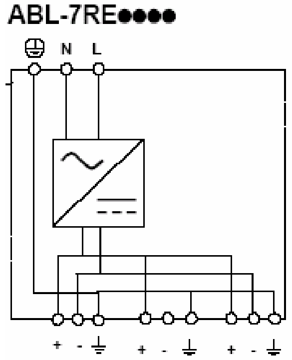
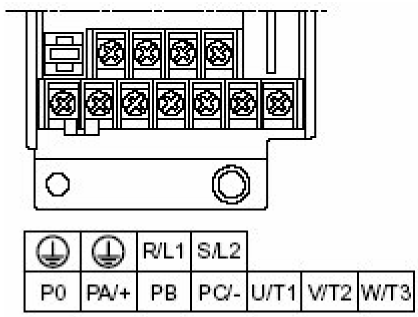
General

- The components designed for installation in a control cabinet, e.g., Phaseo power supply units, Advantys I/O islands, safety modules, line circuit breakers, contactors, and motor circuit breakers, can be snapped onto a 35 mm top-hat rail.
- Master switches, Altivar variable speed drives and Lexium servo drives are screwed directly onto the mounting plate.
- Emergency-off, switches and housings for display and acknowledge indicators are designed for backplane assembly in the field; all switches can also be installed directly in a control cabinet (e.g., on control cabinet door) without their enclosing housings.
- There are two options for installing XB5 pushbuttons or indicator lamps: These pushbuttons or switches can be installed either in a 22 mm hole, e.g., drilled into the front door of the control cabinet, or in an XALD-type housing suitable for up to 5 pushbuttons or indicator lamps. The XALD switch housing is designed for backplane assembly or direct wall mounting.
- The operator and display terminal requires a cutout to be made in the front of the housing. It is then attached to the housing by means of screwed brackets.
- 400 V/3-phase AC wiring for the load circuits (ATV, LXM)
- 240 V AC wiring for the power supplies.
- 24 V DC wiring for the ATV/LXM control circuits and PLC/HMI power supply

The CANopen bus lines are used for the communication link between the PLC and the variable speed drives, servo drives and I/O island in the main cabinet.

Mains Switch **NSC 100 Compact**		
Emergency OFF Switch (tamper free) **XALK178G**		

Selector and pushbutton switch **XB5**	 	 
Preventa safety relay **XPS-AF5130**		
Motor circuit breaker for 400 V AC for LEX05 and ATV71 **GV2ME16**		
Motor circuit breaker for 400 V AC for ATV31 circuit breaker (short-circuit protection) **GV2-L**		
Motor contactor **LC1D**		

Back-up fuse for 24 V DC **GB2-DB07**		
Power supply **ABL7RE2405**		
ATV31 variable speed drive incl. line supply and motor connection **ATV31H037N4**		

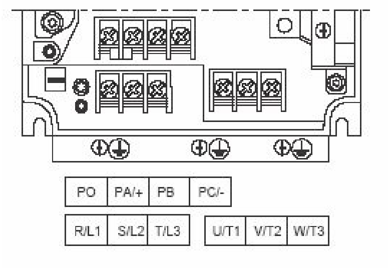
Description of terminals:

Terminal	Function
⏚	Ground terminal
R/L1 S/L2	Power supply
R/L1 S/L2 T/L3	
PO	DC bus + polarity
PA/+	Output to braking resistor (+ polarity)
PB	Output to braking resistor
PC/-	DC bus - polarity
U/T1 V/T2 W/T3	Outputs to the motor

ATV71 variable speed drive incl. line supply and motor connection

ATV71H075N4



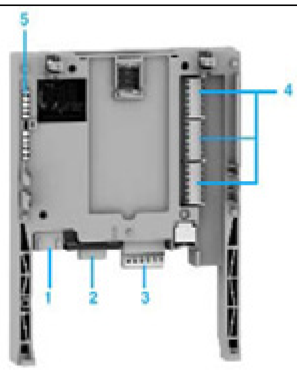


Description of terminals:

Terminal	Function
⏚	Ground terminal
R/L1 S/L2	Power supply
R/L1 S/L2 T/L3	
PO	DC bus + polarity
PA/+	Output to braking resistor (+ polarity)
PB	Output to braking resistor
PC/-	DC bus - polarity
U/T1 V/T2 W/T3	Outputs to the motor

ATV71 variable speed drive
Controller Inside card

VW3 A3501

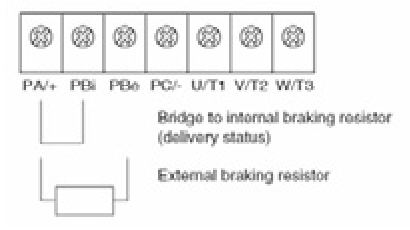


1. RJ45 connector for the PS 1131 programming software
2. CANopen bus
3. Connector for the 24 V power supply and 4 logic inputs
4. Connector for 6 logic inputs, 6 logic outputs, 2 analog inputs and 2 analog outputs and 2 common connections
5. 5 signaling LEDs

Lexium05 servo drive 3-phase

LXM05AD14N4



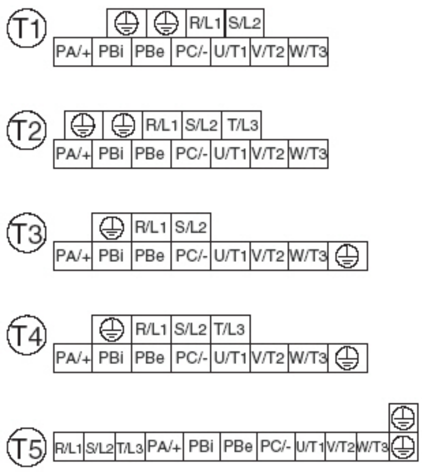


Power connections	Description
PE	Earth connection (protective earth)
R/L1, S/L2/N	Mains connection, single phase devices
R/L1, S/L2, T/L3	Mains connection, 3-phase devices
PA/+	DC bus
PBi	Braking resistor internal
PBe	Braking resistor external
PC/-	DC bus
U/T1,V/T2, W/T3	Motor connections

Lexium05 servo drive 3-phase

LXM05AD14N4

Power terminals see T4

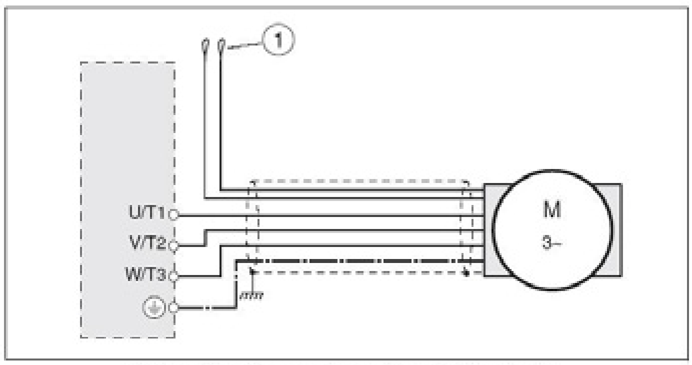


LXM05•...	
D10F1	(T1)
D10M2	(T1)
D10M3X	(T2)
D14N4	(T4)
D17F1	(T3)
D17M2	(T4)
D17M3X	(T4)
D22N4	(T4)
D28F1	(T3)
D28M2	(T4)
D34N4	(T4)
D42M3X	(T4)
D57N4	(T5)

Lexium05 servo drive 3-phase

LXM05AD14N4

Motor phase connection



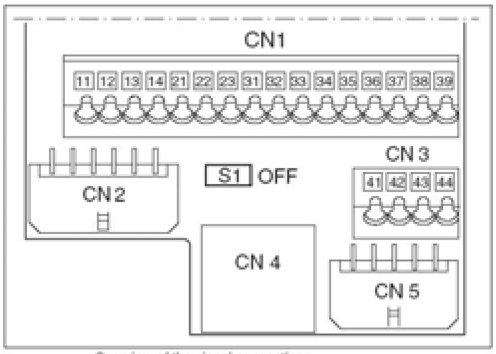
Motor wiring diagram, here without holding brake

Terminal	Description	Colour
U/T1	Motor lead	black L1 (BK)
V/T2	Motor lead	black L2 (BK)
W/T3	Motor lead	black L3 (BK)
PE	Protective conductor	green/yellow (GN/YE)
(1)	Holding brake connection cable For motors with holding brake	white (WH), grey (GR)

Lexium05 servo drive 3-phase

LXM05AD14N4

Signal terminals



Overview of the signal connections

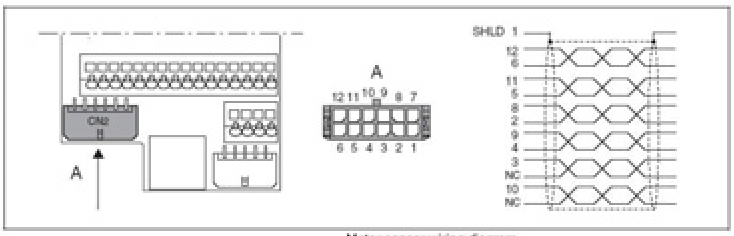
Connection/ switch	Assignments
CN1	Analogue inputs ±10V, pin 11 to 14
	CANopen, pin 21-23
	Digital inputs/outputs, pin 31-39
CN2	Motor encoder (Hiperface Sensor)
CN3	24V PELV controller supply voltage
CN4	PC, peripheral operating terminal, Modbus, CANopen;(RJ45)
CN5	ESIM (A/B/I out), PULSE/DIR in, encoder signals A/B/I in [1]
S1	Switch for fieldbus terminating resistor

1) depending on the "First Setup"

Lexium05 servo drive 3-phase

LXM05AD14N4

Motor encoder



Motor sensor wiring diagram

Pin	Signal	Motor, pin	Colour [1]	Pair	Description	I/O
1	SHLD				Shielding braid	
12	SIN	8	white	1	Sine signal	E
6	REFSIN	4	brown	1	Reference for sine signal, 2.5 V	A
11	COS	9	green	2	Cosine signal	E
5	REFCOS	5	yellow	2	Reference for cosine signal, 2.5V	A
8	Data	6	grey	3	Receive and transmit data	I/O
2	$\overline{\text{Data}}$	7	pink	3	Receive and transmit data, inverted	I/O
10	ENC_0V	11	blue	4	sensor reference potential (encoder) (0.5mm²)	A
			red	4	not assigned (0.5mm²)	
3	TMOT_0V	1	black	5	Reference potential for T_MOT	-
			purple	5	not assigned	
9	T_MOT	2	grey/pink	6	temperature sensor PTC	E
4	ENC+10V_OUT	10	red/blue	6	10 V_{DC} power supply for sensor, max. 150 mA	A
7	n.c.				not assigned	

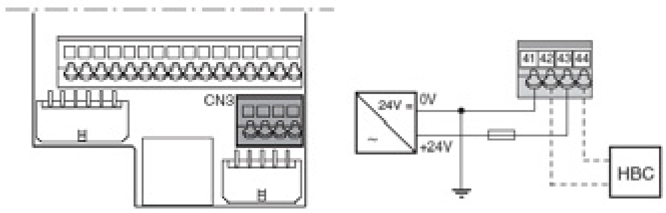
1) Colour data is based on the prefabricated cables

Lexium05 servo drive 3-phase

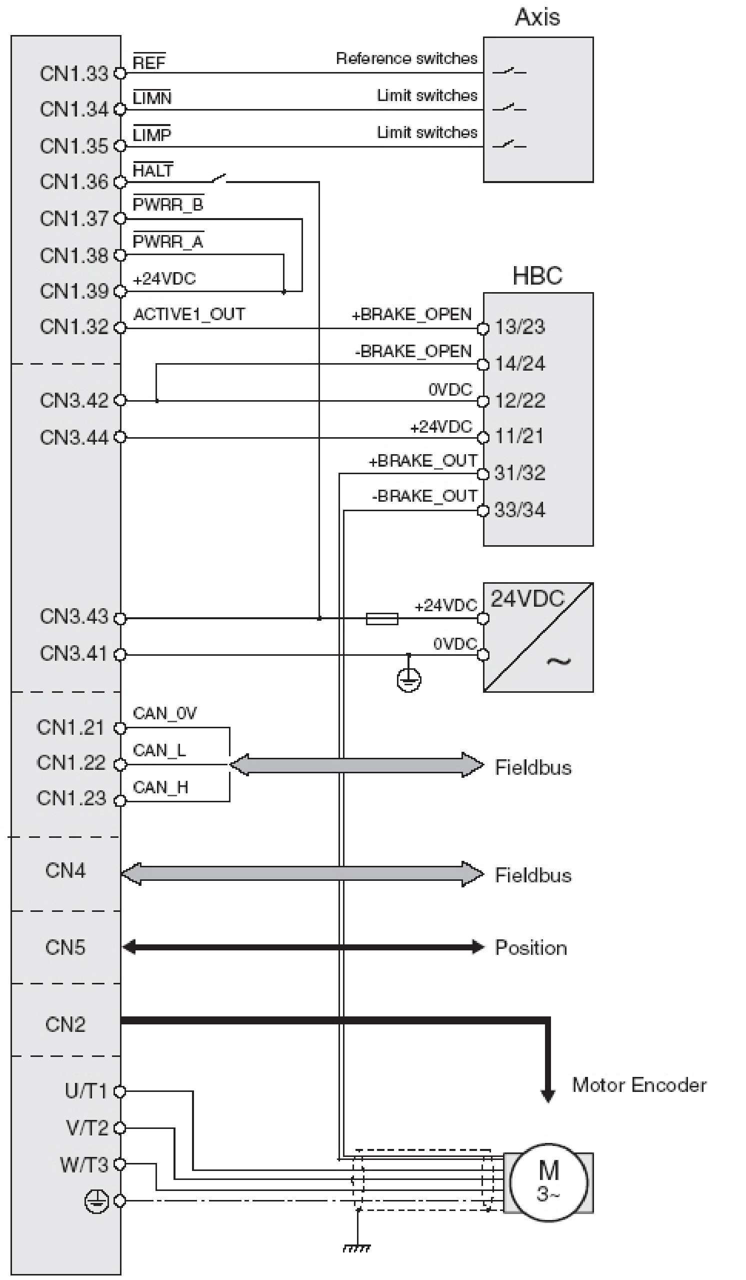
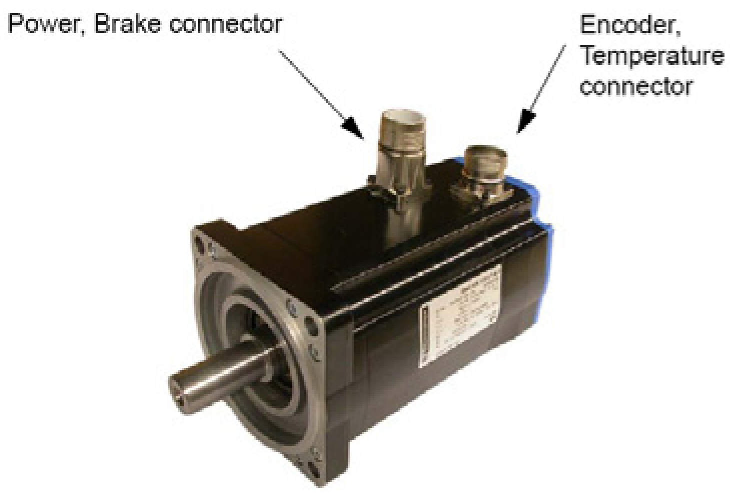
LXM05AD14N4

Control power supply

HBC = Holding Brake Control

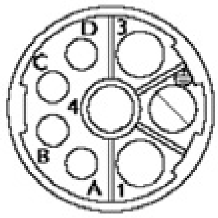


Pin	Signal	Description
41	0VDC	Reference potential for 24V voltage
42	0VDC	Reference potential for 24V voltage
43	+24VDC	24V controller supply voltage
44	+24VDC	24V controller supply voltage

Lexium05 servo drive 3-phase **LXM05AD14N4** Fieldbus wiring Terminals 33-39 must be connected. The motor brake (if present) must be connected via a holding brake control (HBC) We use the RJ45 terminal for CANopen.	
Servo motor **BSH0702P31A2A** **(old:SER39B4L3SSOAOO)**	

Power cable 3 m

VW3 M5 101R30
(old:GEA2MOAAAA003)



Pin	Designation (lead no.)	Meaning	Range
1	U	power	3 AC 0 - 480V
2	PE	shield	
3	W	power	3 AC 0 - 480V
4	V	power	3 AC 0 - 480V
A	brake +	brake	DC 24V
B	brake -	brake	DC 0V
C	-	not assigned	
D	-	not assigned	

Feedback cable 3 m

VW3 M8 101R30
(old:GEA 2EAAAAA003)

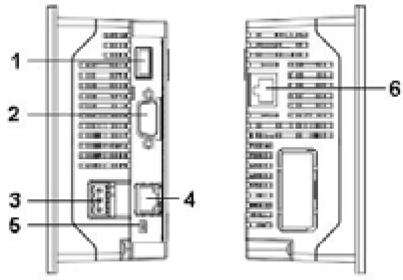


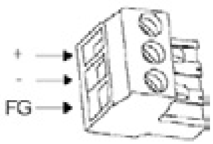
Pin	Designation (lead no.)	Meaning	Range
1	Sensor PTC	Temperature	
2	Sensor PTC	Temperature	
3	-	not assigned	
4	REF SIN	REF signal	
5	REF COS	REF signal	
6	Data +	RS 485	
7	Data -	RS 485	
8	SIN +		
9	COS +		
10	U	power	DC 7 - 12V
11	GND	Ground	DC 0V
12	-	not assigned	

Magelis HMI

XBT-GT 2330



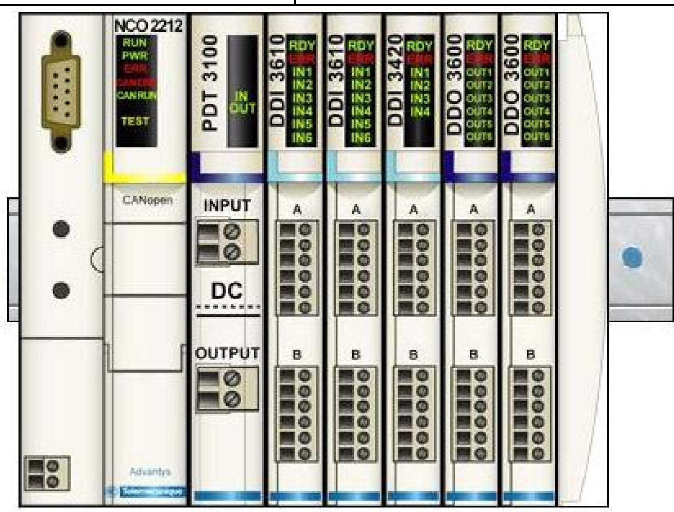


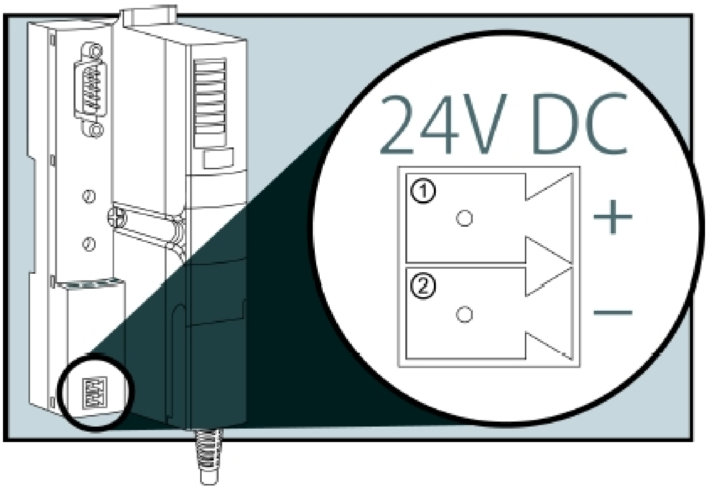
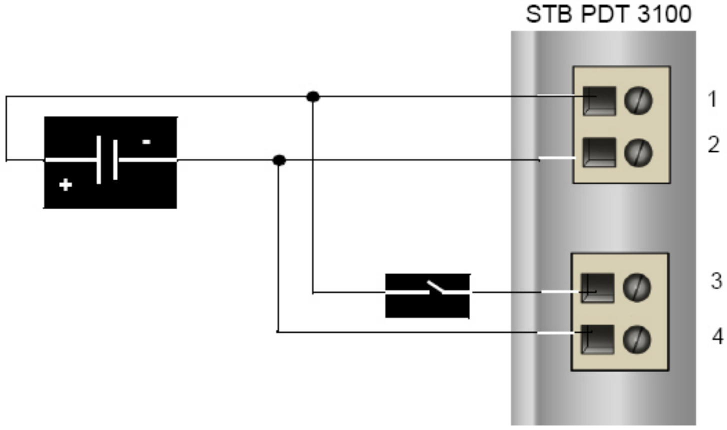


+	24 VDC
-	0 V
FG	Earth

1 USB-Port (USB1.1)
2 Serial Port COM1 (SubD, 9-polig)
3 Power Connection (see left)
4 Serial Port COM2 (RJ45)
5 Polarisation selector
6 Ethernet connector (10BASE-T/100BASE-TX)

Advantys STB



Advantys STB **CANopen bus adapter STB NCO 2212**	 **1** receptacle 1—24 VDC **2** receptacle 2—common voltage
Advantys STB **Field power supply** **STB PDT 3100** **Note:** The output power supply can be deactivated independently of the inputs (e.g., in case of an emergency stop).	 **1** +24 VDC sensor bus power **2** -24 VDC sensor power return **3** +24 VDC actuator bus power **4** -24 VDC actuator power return

Software

General

Software is primarily used for two reasons, first for programming the Premium PLC and configuring CANopen communication and second for generating visualization.

The PLC is programmed using the PS1131 programming tool (CoDeSys).

The HMI application on the XBT-GT 2330 Magelis display terminal is configured using the Vijeo Designer software.

The Altivar 71 and 31 variable speed drives and the Lexium05 servo drive can be parameterized via the front operator panel. However, using the PowerSuite software is much easier. As well as providing a convenient means of setting speed-drive/servo parameters, this software also enables data to be saved and archived. These functions are extremely useful as they mean that parameters can be restored rapidly whenever service tasks need to be performed. The software can also help you to optimize the parameters online.

The Advantys Configuration Tool software is used to parameterize I/O islands.

To use the software packages, your PC must have the appropriate Microsoft Windows operating system installed:

- Windows 2000 *or*
- Windows XP

The software tools have the following default install paths:



- PS1131 (CoDeSys)
  C:\Program Files\Schneider Electric\TwidoSoft
- Vijeo Designer
  C:\Program Files\Schneider Electric\VijeoDesigner
- Advantys Configuration Tool
  C:\Program Files\Schneider Electric\Advantys\
- PowerSuite for e.g., ATV31,ATV71,LXM05
  C:\Program Files\Schneider Electric\PowerSuite







Communication

General

The following methods of communication are used between devices:

- CANopen
- Modbus

The machine bus enabling communication between the PLC and fieldbus devices is implemented in the form of **CANopen**.

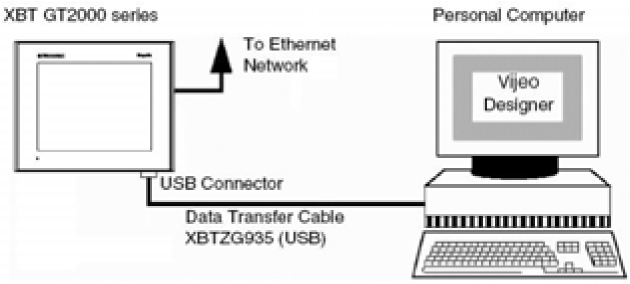
Modbus is used for data exchange between the PLC and remote HMI (Magelis XBTG).

Connection cables are also required between the PC and the individual devices (for programming/parameterization).

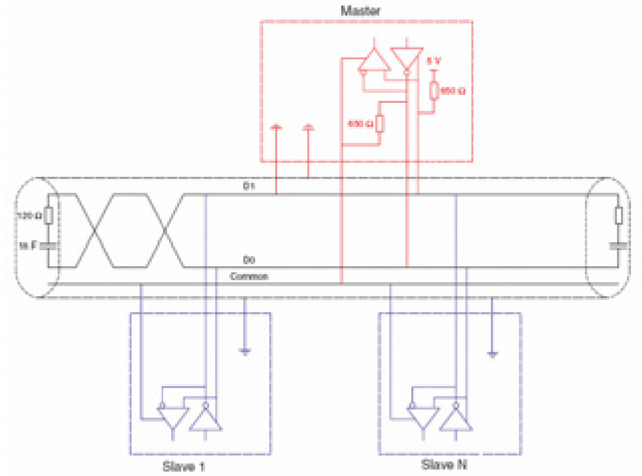
<table>
<tr>
<td>Magelis HMI

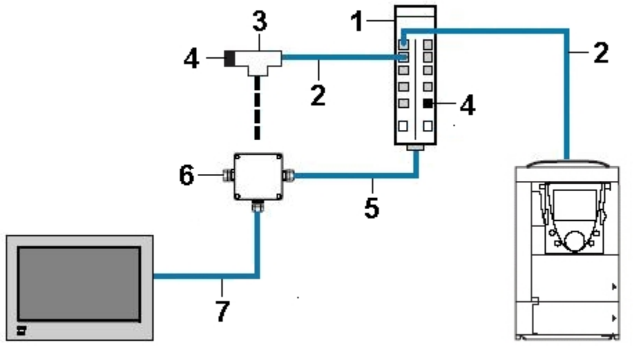
PC – Connection Cable XBTZG935

To download the Configuration from the PC to the HMI

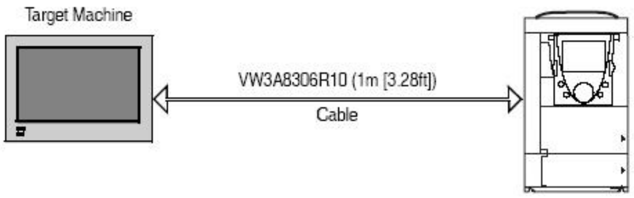
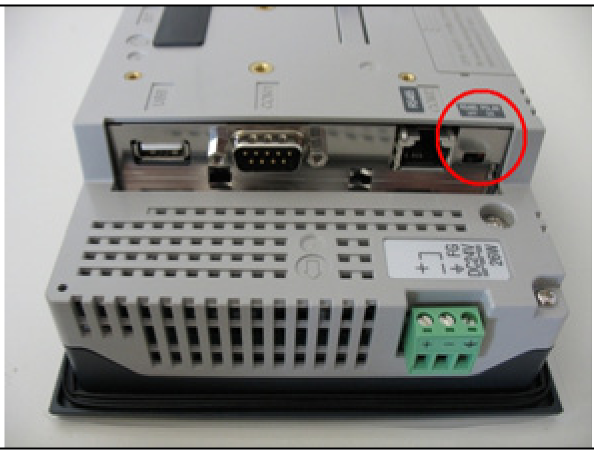
Alternatively you can use the ethernet connection for the configuration.</td>
<td>
</td>
</tr>
<tr>
<td>Modbus

The standard set up can be found on the website Modbus.org in the Modbus specification

(Modbus_over_serial_line_V1.pdf, Nov 2002)
.</td>
<td>
</td>
</tr>
</table>

Modbus Connection Between HMI (COM2) and ATV71. Depending on topography and what components are used.	

1	**LU9 GC3**	**Modbus Hub**
2	**VW3 A8 306 R03**	**Modbus Cable (0,3 m) 2xRJ45**
	VW3 A8 306 R10	**Modbus Cable (1,0 m) 2xRJ45**
	VW3 A8 306 R30	**Modbus Cable (3,0 m) 2xRJ45**
3	**VW3 A8 306 TF03**	**Modbus T-Junction with Cable (0,3 m)**
	VW3 A8 306 TF10	**Modbus T-Junction with Cable (1,0 m)**
4	**VW3 A8 306 RC**	**Terminal resistor**
5	**TSX CSA 100**	**Modbus-Cable (100 m)**
6	**TSX SCA 50**	**Modbus TAP with Terminal Resistor**
7	**VW3 A8 306 D30**	**Modbus-Cable (3,0 m) 1xRJ45**

Connecting Modbus with VW3A8306Rxx For short distances you can use a direct connection between the HMI(COM2) and ATV71. This is, however, not in line with the Modbus specification. If you have problems, use one of the methods listed above.	
Modbus Turn the polarisation on, on the HMI. Set the **Polar** switch to **ON**	

Note

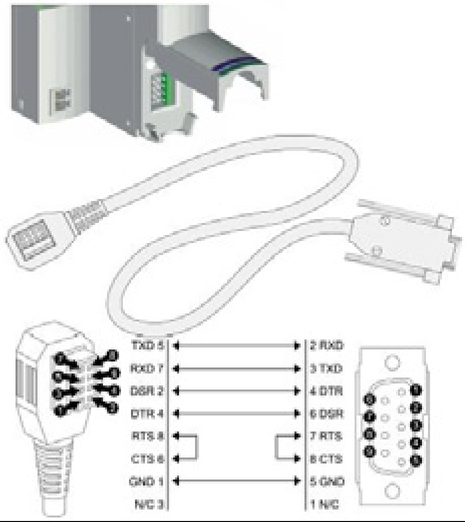
The ATV71 must use the RJ45 port and **NOT** the Controller Inside card port!



Advantys STB

PC connection cable STB XCA 4002

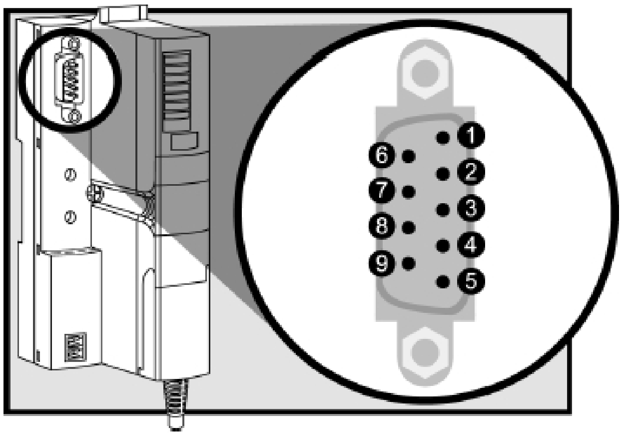
This connection cable is supplied with the Advantys Configuration Software.



Advantys STB

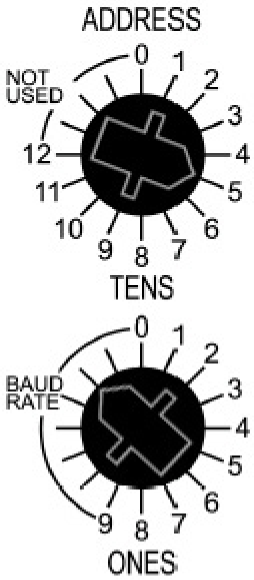
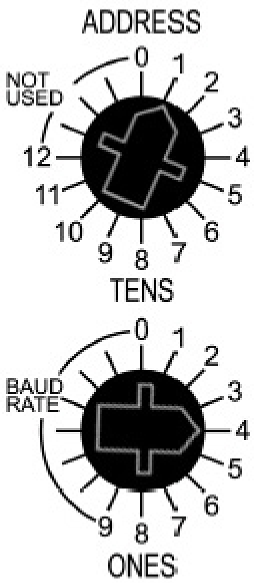
CANopen bus adapter STB NCO 2212

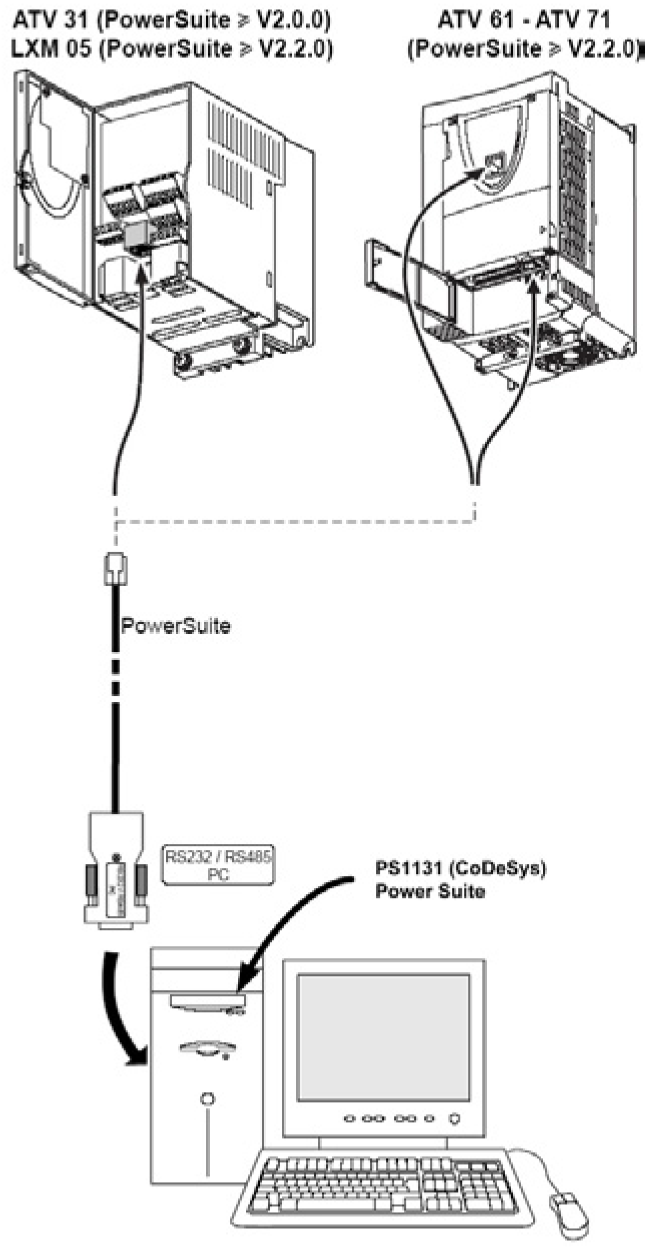
CANopen fieldbus port



Pin	Signal	Description
1	Unused	Reserved
2	CAN_L	CAN-low bus line
3	CAN_GND	CAN ground
4	Unused	Reserved
5	CAN_SHLD	optional CAN shield
6	GND	optional ground
7	CAN_H	CAN-high bus line
8	Unused	Reserved
9	Unused	Reserved

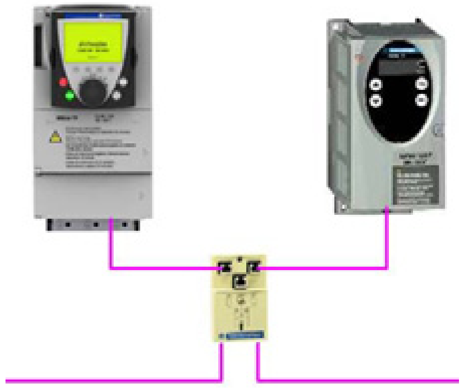
Note: Pin numbers correspond to callouts in the figure above.

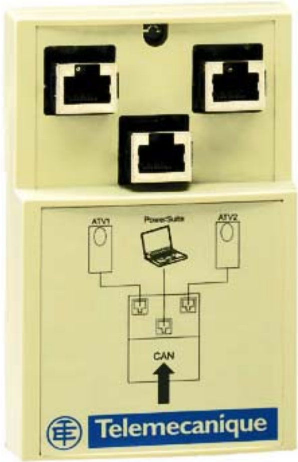
Advantys STB **CANopen bus adapter STB NCO 2212** CANopen baud rate 	The rotary switches on the STB NCO 2212 CANopen NIM are used to set the network node address and the Advantys STB island's **baud rate**. 1. Disconnect the island's power supply. 2. Set the lower rotary switch to any position between 9 and 0 (baud rate setting is marked). 3. Select the baud rate you wish to use for fieldbus communication. Select an appropriate baud-rate setting for your system and network requirements. 0 - 10,000 bps 4 - 250,000 bps 1 - 20,000 bps **5 - 500,000 bps** 2 - 50,000 bps 6 - 800,000 bps 3 - 125,000 bps 7 - 1 Mbps In this example we have selected setting "5" (500,000 bps). 4. Turn the upper rotary switch to the position corresponding to the baud rate you have selected (e.g., "5"). 5. Power up your island to use the new settings. The NIM only reads the rotary-switch settings on power-up.
Advantys STB **CANopen bus adapter STB NCO 2212** CANopen address 	The rotary switches on the STB NCO 2212 CANopen NIM are used to set the **network node address** and the Advantys STB island's baud rate. 1. Be sure to set the required baud rate (following the procedure described above) before setting the network node address. 2. Disconnect the island's power supply. 3. Select a network node address that is available in your fieldbus network. 4. Set the lower rotary switch to the position corresponding to the one's place of the required node address. For network node address 14, the lower switch would be set to **4**. 5. Set the upper rotary switch to the position corresponding to the ten's and hundred's place of the required node address. For network node address 14, the upper switch would be set to **1**. 6. Switch on Advantys STB. The NIM only reads the rotary-switch settings on power-up.

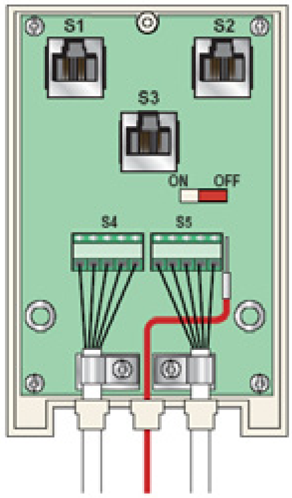
Connection cable set for PowerSuite and PS1131 (CoDeSys) **VW3 A8106** You will need the RS232 to RS485 adapter and the PowerSuite cable for the connection between the PC and the PowerSuite software and VSDs/servos. You will also need the adapter and the cable for the PS1131 software. The cable must be disconnected and reconnected to the programming port on the Controller Inside card installed in the ATV71.	

CANopen junction box

VW3 CAN TAP2







Pin	Signal	Wire colour	Description
1	GND	Black	Ground
2	CAN_L	Blue	CAN_L bus line
3	SHLD	(bare cableshield)	Optional shield
4	CAN_H	White	CAN_H bus line
5	(V+)	Red	Optional supply

For this application example, the slide switch must be set to **OFF**.

If, unlike in this application, the junction box does not have an outgoing CANopen bus, the line terminator must be activated (i.e., set to ON).

CANopen preassembled connection cable

VW3 CAN CARRxx

This cable is used to connect the junction box to the ATV31, ATV71 and LXM05.



VW3 CAN CARR1 (length: 1.0 m)



VW3 CAN CARR03 (length: 0.3 m)

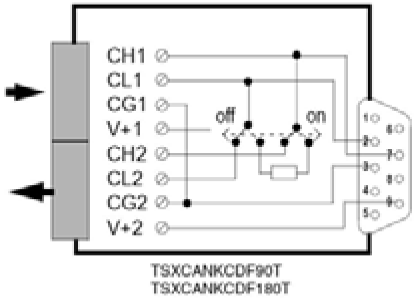
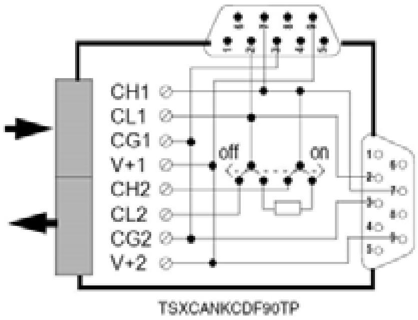
CANopen connector

VW3CANKCDF180T

This connector is used for the link to the CANopen master (Controller Inside card in the ATV71).



At the **start of the bus**, the terminating resistor must be active. To do this, set the switch to **ON**. The bus cable must be connected on the incoming side.

CANopen connector **VW3 CAN KCDF 90TP** This connector is used for the link to the Advantys STB I/O island.		At the **end of the bus**, the terminating resistor must be activated. To do this, set the switch to **ON**.
CANopen connector assignment **VW3 CAN KCDF xxxx**	 	
CANopen cable **TCX CAN Cx yy** The cable is available in various versions (**x**): Standard No Flame Heavy Duty and various lengths (**yy**): 50,100, 300 m.		

Signal	Terminal block 1	Terminal block 2	Wire color
CAN_H	CH1	CH2	white
CAN_L	CL1	CL2	blue
CAN_GND	CG1	CG2	black
CAN_V+	V+1	V+2	red

Implementation

Introduction

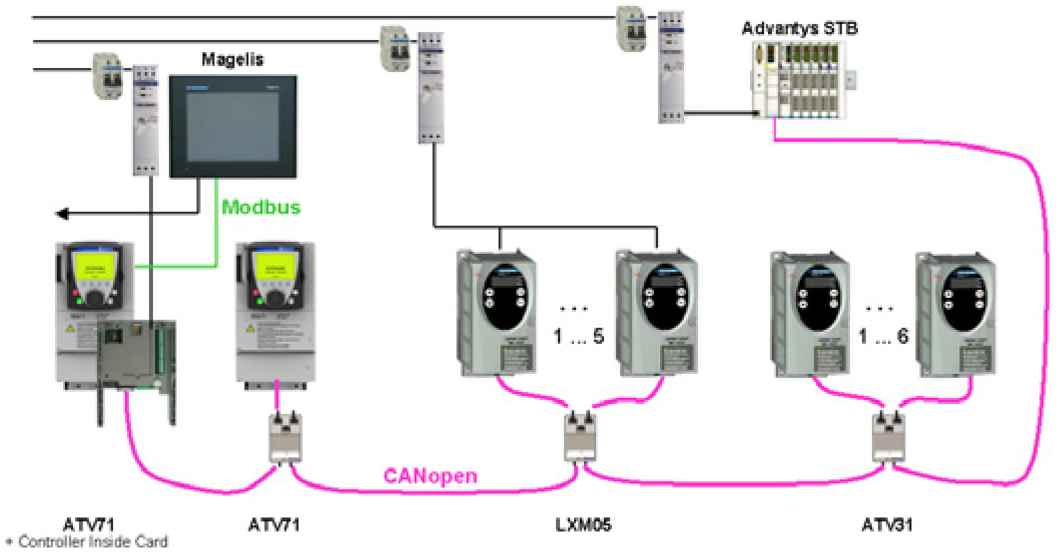
The implementation chapter describes all the steps necessary to initialize, to configure, to program and start-up the system to achieve the application functions as listed below.

Function

Instructions for switching on and functional description

1. Switch on all fuses and motor circuit breakers.
2. Switch on the master switch.
3. Acknowledge emergency-off signals.
4. Acknowledge error messages to HMI.
5. You can now select MANUAL or AUTOMATIC mode on the main screen.
6. Manual: On the ATV31, AVT71 and Lexium screen, the individual motors can be moved independently of one another.
7. Automatic: No applications are active here.
8. The BUS display indicates the states of the individual CANopen nodes.

Functional Layout



Communication

Introduction

This chapter describes the data passed via the communications bus (e.g. Modbus Plus or CANopen) that is not bound directly with digital or analog hardware.

The list contains:

- The device links
- Direction of data flow
- symbolic name and
- Bus address of the device concerned.

Device Links

The Modbus and CANopen bus systems are used in this application.

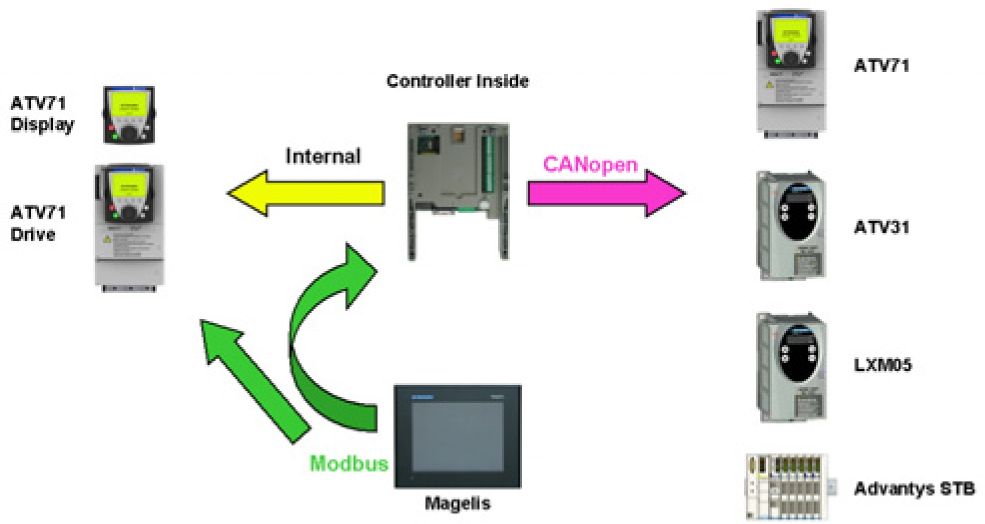
The following devices are networked via **CANopen**:

- An Altivar 71 with Controller Inside – PLC, bus address 1 (master)
- An Altivar 71 variable speed drive, bus address 2
- Five Lexium05 servos, bus addresses 3 to 7
- Six Altivar 31 variable speed drives, bus addresses 8 to 13
- One Advantys STB I/O island, bus address 14

Two devices are interconnected via **Modbus**:

- Magelis panel XBT-GT
- Altivar 71 with Controller Inside PLC, bus addresses 2 + 8

The Controller Inside PLC and the variable speed drive can be addressed separately via different Modbus addresses over the same physical interface on the Altivar.



The following **CANopen** settings are used in this application:

CANopen Addresses

Controller Inside PLC is *CANopen master*	
Device	**CANopen Address**
Controller Inside	1
1. Altivar 71	none
2. Altivar 71	2
1. Lexium 05	3
2. Lexium 05	4
3. Lexium 05	5
4. Lexium 05	6
5. Lexium 05	7
1. Altivar 31	8
2. Altivar 31	9
3. Altivar 31	10
4. Altivar 31	11
5. Altivar 31	12
6. Altivar 31	13
Advantys STB	14

A baud rate of 500 kbps is used.

The settings for the heartbeat used to monitor the nodes are as follows:

Master 300 ms (PLC -> slaves)
Slaves 200 ms (slaves -> PLC)

CANopen COB-ID

Data Direction Device --> PLC (TPDO)						
Device	**1.PDO**	**2.PDO**	**3.PDO**	**4.PDO**	**5.PDO**	**6.PDO**
2. ATV71	182	---	---			
1. LXM05	183	---	---	483		
2. LXM05	184	---	---	484		
3. LXM05	185	---	---	485		
4. LXM05	186	---	---	486		
5. LXM05	187	---	---	487		
1. ATV31	---					698
2. ATV31	---					699
3. ATV31	---					69A
4. ATV31	---					69B
5. ATV31	---					69C
6. ATV31	---					69D
Advantys STB	18E	28E				
Data direction PLC --> Device (RPDO)						
Device	**1.PDO**	**2.PDO**	**3.PDO**	**4.PDO**	**5.PDO**	**6.PDO**
2. ATV71	202	---	---			
1. LXM05	203	---	---	503	683	
2. LXM05	204	---	---	484	684	
3. LXM05	205	---	---	485	685	
4. LXM05	206	---	---	486	686	
5. LXM05	207	---	---	487	687	
1. ATV31	---					688
2. ATV31	---					688
3. ATV31	---					68A
4. ATV31	---					68B
5. ATV31	---					68C
6. ATV31	---					68D
Advantys STB	20E					

Datalink
PLC <> ATV71

Controller Inside *(CANopen-Master)*			Altivar 71 *(CANopen-Slave)*	
Data direction PLC <-- ATV (TPDO)				
Address	**2.ATV**	**Designation**	**Index**	**Designation**
%IW	80	direct to EFB input	6041	Drivecom status register
%IW	81	direct to EFB input	6044	Control effort
%IW	82	direct to EFB input	603F	Error Code
%IW	83	not used	./.	Reserve
%IW	84	not used	./.	Reserve
%IW	85	not used	./.	Reserve
%IW	86	not used	./.	Reserve
Data direction PLC --> ATV (RPDO)				
Address	**2.ATV**	**Designation**	**Index**	**Designation**
%QW	80	direct from EFB output	6040	Drivecom command reg.
%QW	81	direct from EFB output	6042	Target velocity
%QW	82	not used	./.	Reserve
%QW	83	not used	./.	Reserve
%QW	84	not used	./.	Reserve
%QW	85	not used	./.	Reserve

Datenlink
PLC <> LXM05

Controller Inside *(CANopen-Master)*							Lexium 05 *(CANopen-Slave)*	
Data Direction PLC <-- LXM (TPDO)								
Address	**1.LXM**	**2.LXM**	**3.LXM**	**4.LXM**	**5.LXM**		**Index**	**Designation**
%IW	91	107	123	139	155		6041	Drivecom status register
%IW	92	108	124	140	156		./.	Reserve
%IW	93	109	125	141	157		./.	Reserve
%IW	95	111	127	143	159		./.	Reserve
%ID	48	56	64	72	80		./.	Reserve
%ID	49	57	65	73	81		606B	Velocity actual value
%ID	50	58	66	74	82		6064	Position actual value
Data Direction PLC --> LXM (RPDO)								
Address	**1.LXM**	**2.LXM**	**3.LXM**	**4.LXM**	**5.LXM**		**Index**	**Designation**
%QW	91	107	123	139	155		6040	Drivecom command reg.
%QW	92	108	124	140	156		./.	Reserve
%QW	93	109	125	141	157		./.	Reserve
%QW	95	111	127	143	159		./.	Reserve
%QD	48	56	64	72	80		./.	Reserve
%QD	49	57	65	73	81		60FF	Target velocity
%QD	50	58	66	74	82		607A	Target position
%QD	51	59	67	75	83		6081	Profile velocity

Datalink
PLC <> ATV31

Controller Inside *(CANopen-Master)*							Altivar 31 *(CANopen-Slave)*	
Data Direction PLC <-- ATV (TPDO)								
Address	**1.ATV**	**2.ATV**	**3.ATV**	**4.ATV**	**5.ATV**	**6.ATV**	**Index**	**Designation**
%IW	170	175	180	185	190	195	./.	Reserve
%IW	171	176	181	186	191	196	6041	Drivecom status register
%IW	172	177	182	187	192	197	6044	Control effort
%IW	173	178	183	188	193	198	603F	Error Code
Data Direction PLC --> ATV (RPDO)								
Address	**1.ATV**	**2.ATV**	**3.ATV**	**4.ATV**	**5.ATV**	**6.ATV**	**Index**	**Designation**
%QW	170	175	180	185	190	195	./.	Reserve
%QW	171	176	181	186	191	196	6040	Drivecom command reg.
%QW	172	177	182	187	192	197	6042	Target velocity

Datalink PLC <> STB

Controller Inside *(CANopen-Master)*			Advantys *(CANopen-Slave)*	
Data Direction PLC <-- STB (TPDO)				
Address	**Bit**	**Designation**	**Input word**	**Designation**
%IB400	0...5	1. Input Module, Input 1...6	1	Slot 3 - Input 1...6
%IB401	0...5	1. Input Module, Status 1...6	1	Slot 3 - Status 1...6
%IB402	0...5	2. Input Module, Input 1...6	2	Slot 4 - Input 1...6
%IB403	0...5	2. Input Module, Status 1...6	2	Slot 4 - Status 1...6
%IB404	0...3	3. Input Module, Input 1...4	3	Slot 5 - Input 1...4
%IB404	4...7	3. Input Module, Status 1...4	3	Slot 5 - Status 1...4
%IB405	0...5	1. Output Module, Echo 1...6	3	Slot 6 - Echo 1...6
%IB406	0...5	1. Output Module, Status 1...6	4	Slot 6 - Status 1...6
%IB407	0...5	2. Output Module, Echo 1...6	4	Slot 7 - Echo 1...6
%IB408	0...5	2. Output Module, Status 1...6	5	Slot 7 - Status 1...6
Data Direction PLC --> STB (RPDO)				
Address	**Bit**	**Designation**	**Output word**	**Designation**
%QB400	0...5	1. Output Module, Output 1...6	1	Slot 6 - Output 1...6
%QB401	0...5	2. Output Module, Output 1...6	1	Slot 7 - Output 1...6

The following **Modbus** settings are used in this application:

Modbus Addresses

Magelis HMI is *Modbus slave*		
Device	**Modbus Address**	**Interface**
Magelis HMI	1	COM2
Controller Inside	8	RJ45 on ATV71
1. Altivar 71	2 not used	RJ45
2. Altivar 71	1 not used	RJ45
1. - 6. Altivar 31	1 not used	RJ45
1. - 5. Lexium05	1 not used	RJ45

A baud rate of 19.2 kbps is used.
There are 8 data bits, 1 stop bit and even parity.

There is only a Modbus link between the HMI and the Controller Inside cards.

Datalink HMI <> PLC for Altivar

Controller Inside								Magelis HMI
Data Direction HMI <--> PLC (for Altivar)								
Name	**Type**	**Addr.**	**2. ATV71**	**1. ATV31**	**2. ATV31**	**3. ATV31**	**Bit**	**Designation**
Start	BOOL	%MW	1000	1010	1020	1030	0	Start Command
Estop	BOOL						8	emergency Off
Dir	BOOL	%MW	1001	1011	1021	1031	0	direction of Revolutions
Ackn	BOOL						8	Acknowledgement
Error	BOOL	%MW	1002	1012	1022	1032	0	Error message
CommOK	BOOL						8	Communication OK
Run	BOOL	%MW	1003	1013	1023	1033	0	running
Mot_ES	BOOL						8	Motor Emergency off
AC_pwr_OK	BOOL	%MW	1004	1014	1024	1034	0	Power OK
Res	BOOL						8	Reserve
Speed_Set	WORD	%MW	1005	1015	1025	1035	---	Set Revolutions
Speed_Act	WORD	%MW	1006	1016	1026	1036	---	Actual Revolutions
ErrCode	WORD	%MW	1007	1017	1027	1037	---	Errorcode
CANopen	WORD	%MW	1008	1018	1028	1038	---	Status Communication
Data Direction HMI <--> PLC (for Altivar)								
Name	**Type**	**Addr.**	**4. ATV31**	**5. ATV31**	**6. ATV31**	**1. ATV71**	**Bit**	**Designation**
Start	BOOL	%MW	1040	1050	1060	1070	0	Start Command
Estop	BOOL						8	Emergency Off
Dir	BOOL	%MW	1041	1051	1061	1071	0	Direction of Revolutions
Ackn	BOOL						8	Acknowledgement
Error	BOOL	%MW	1042	1052	1062	1072	0	Error Message
CommOK	BOOL						8	Communication OK
Run	BOOL	%MW	1043	1053	1063	1073	0	Running
Mot_ES	BOOL						8	Motor Emergency Off
AC_pwr_OK	BOOL	%MW	1044	1054	1064	1074	0	Power OK
Res	BOOL						8	Reserve
Speed_Set	WORD	%MW	1045	1055	1065	1075	---	Set Revolutions
Speed_Act	WORD	%MW	1046	1056	1066	1076	---	Actual Revolutions
ErrCode	WORD	%MW	1047	1057	1067	1077	---	errorcode
CANopen	WORD	%MW	1048	1058	1068	1078	---	Status Communication

Datalink HMI <> PLC for Lexium

Controller Inside						Magelis HMI	
Data direction HMI <--> PLC (for Lexium05)							
Name	**Type**	**Address**	**1.LXM**	**2.LXM**	**3.LXM**	**Bit**	**Designation**
Start	BOOL	%MW	1100	1122	1144	0	Start Command
Estop	BOOL					8	Emergency off
Dir	BOOL	%MW	1101	1123	1145	0	Direction of Rotation
Ackn	BOOL					8	Acknowledgement
ModeOK	BOOL	%MW	1102	1124	1146	0	Operating Mode OK
Mode_T	BOOL					8	Operating Mode Selectior
CommOK	BOOL	%MW	1103	1125	1147	0	Communication OK
PosOK	BOOL					8	Position reached
Run	BOOL	%MW	1104	1126	1148	0	Power OK
Mot_ES	BOOL					8	Motor running
Error	BOOL	%MW	1105	1127	1149	0	Error Message
Res	BOOL					8	Reserve
Speed_Set	DINT	%MW	1106	1128	1150	---	Set Revolutions
		%MW	1107	1129	1151	---	
Speed_Act	DINT	%MW	1108	1130	1152	---	Actual Revolutions
		%MW	1109	1131	1153	---	
Position_Set	DINT	%MW	1110	1132	1154	---	Set Position
		%MW	1111	1133	1155	---	
Position_Act	DINT	%MW	1112	1134	1156	---	Actual Position
		%MW	1113	1135	1157	---	
Profil_spd	DINT	%MW	1114	1136	1158	---	Profile Speed
		%MW	1115	1137	1159	---	
Node_ID	WORD	%MW	1116	1138	1160	---	CANopen Address
ErrCode	WORD	%MW	1117	1139	1161	---	Errorcode
ErrorC	WORD	%MW	1118	1140	1162	---	Errorcode
CANopen	WORD	%MW	1119	1141	1163	---	Status Communication
SDO_EN	BOOL	%MW	1120	1142	1164	0	SDO enable
SDO_done	BOOL					8	SDO sent
Mode	WORD	%MW	1121	1143	1165	---	operation mode

Datalink HMI <> PLC for Lexium and STB

Controller Inside							Magelis HMI
Data Direction HMI <--> PLC (for Lexium05 and Advantys STB)							
Name	**Type**	**Address**	**4.LXM**	**5.LXM**	**STB**	**Bit**	**Designation**
Start	BOOL	%MW	1166	1188	---	0	Start Command
Estop	BOOL					8	emergency Off
Dir	BOOL	%MW	1167	1189	---	0	Direction of Rotation
Ackn	BOOL					8	Acknowledgement
ModeOK	BOOL	%MW	1168	1190	---	0	Mode OK
Mode_T	BOOL					8	Mode Selection
CommOK	BOOL	%MW	1169	1191	1251	0	Communication OK
PosOK	BOOL				---	8	Position reached
Run	BOOL	%MW	1170	1192	---	0	Resistance OK
Mot_ES	BOOL					8	Run message
Error	BOOL	%MW	1171	1193	---	0	Error message
Res	BOOL					8	Reserve
Speed_Set	DINT	%MW	1172	1194	---	---	Set value Revolutions
		%MW	1173	1195	---	---	
Speed_Act	DINT	%MW	1174	1196	---	---	Actual Revolutions
		%MW	1175	1197	---	---	
Position_Set	DINT	%MW	1176	1198	---	---	Set Value Position
		%MW	1177	1199	---	---	
Position_Act	DINT	%MW	1178	1200	---	---	Actual Position
		%MW	1179	1201	---	---	
Profil_spd	DINT	%MW	1180	1202	---	---	Profile Revolutions
		%MW	1181	1203	---	---	
Node_ID	WORD	%MW	1182	1204	---	---	CANopen Address
ErrCode	WORD	%MW	1183	1205	---	---	Errorrcode
ErrorC	WORD	%MW	1184	1206	---	---	Errorcode
CANopen	WORD	%MW	1185	1207	1250	---	Status Communication
SDO_EN	BOOL	%MW	1186	1208	---	0	SDO enable
SDO_done	BOOL					8	SDO sent
Mode	WORD	%MW	1187	1209	---	---	Operation Mode

The following data points are used to animate the header in the HMI:

Datalink HMI <> PLC General

Controller Inside					Magelis HMI	
Data Direction HMI <--> PLC (for Lexium05 and Advantys STB)						
Name	**Type**	**Address**		**Bit**	**Name**	**Designation**
Pos1E	BOOL	%MX	997	0	Drive3.Pos_equal	1.LXM Desired = Actual position
Pos1A	BOOL	%MX	997	1	Drive3.Pos_add	1.LXM Desired position +
Pos1S	BOOL	%MX	997	2	Drive3.Pos_sub	1.LXM Desired position -
Pos2E	BOOL	%MX	997	8	Drive4.Pos_equal	2.LXM Desired = Actual position
Pos2A	BOOL	%MX	997	9	Drive4.Pos_add	2.LXM Desiredposition +
Pos2S	BOOL	%MX	997	10	Drive4.Pos_sub	2.LXM Desiredposition -
BusCANopen	BOOL	%MX	998	0	Head.CANopen	all CANopen bus Devices present
Safety	BOOL	%MX	998	1	Head.Safety	Safety OK
Alarm	BOOL	%MX	998	2	Head.Alarm	No Error Messages
Ackn	BOOL	%MX	998	3	---	Error Acknowledge
Auto	BOOL	%MX	998	4	Head.Auto	Automatic Mode
Manual	BOOL	%MX	998	5	Head.Manual	Manual Mode

Structures are used to enable data exchange between the PLC and HMI. For this purpose, the following variable names are used in the HMI and PLC. The associated structure names in the HMI are also listed in the table.

Structure name

Drive	PLC	HMI
1st Altivar 71	Drive_01.	Drive_01.
2nd Altivar 71	Drive_02.	Drive_02.
1st Lexium 05	DriveLXM[1].	Drive_03.
2nd Lexium 05	DriveLXM[2].	Drive_04.
3rd Lexium 05	DriveLXM[3].	Drive_05.
4th Lexium 05	DriveLXM[4].	Drive_06.
5th Lexium 05	DriveLXM[5].	Drive_07.
1st Altivar 31	Drive31_01.	Drive_08.
2nd Altivar 31	Drive31_02.	Drive_09.
3rd Altivar 31	Drive31_03.	Drive_10.
4th Altivar 31	Drive31_04.	Drive_11.
5th Altivar 31	Drive31_05.	Drive_12.
6th Altivar 31	Drive31_06.	Drive_13.

General Addressing

Various hardware addresses, as well as flags and flag words, are used in the PLC example program. An overview of these appears below to facilitate orientation.

The addresses of individual storage locations are set using special character strings. The addresses comprise a combination of the percentage sign “%”, an area identifier, a data type and one or more natural numbers, which can be separated by blank spaces.

The following area identifiers are supported:

I	- Input
Q	- Output
M	- Flag

The following data types are supported:

X	- Individual bit
None	- Individual bit
B	- Byte (8 bits)
W	- Word (16 bits)
D	- Double word (32 bits)

Examples:

%QX7.5	- Output bit 7.5
%Q7.5	- Output bit 7.5
%QB7	- Output byte 7
%IW215	- Input word 215
%MD48	- Double-word flag in memory location 48
ivar AT %IW0: WORD;	- Variable declaration with indication of address

The program’s current control configuration will determine whether or not an address is valid.

Defined memory areas may overlap, e.g., memory address %QW80 is the same as %QB160,and %QD40

Note:
Boolean values are written as bytes if an individual bit address is not explicitly specified.

Example: A change in the value of varbool1 AT %QW0 affects the area between QX0.0 and QX0.7.

Note:
If Online Change is used, memory addresses may change. Please remember this when using pointers to addresses.

HMI

Introduction

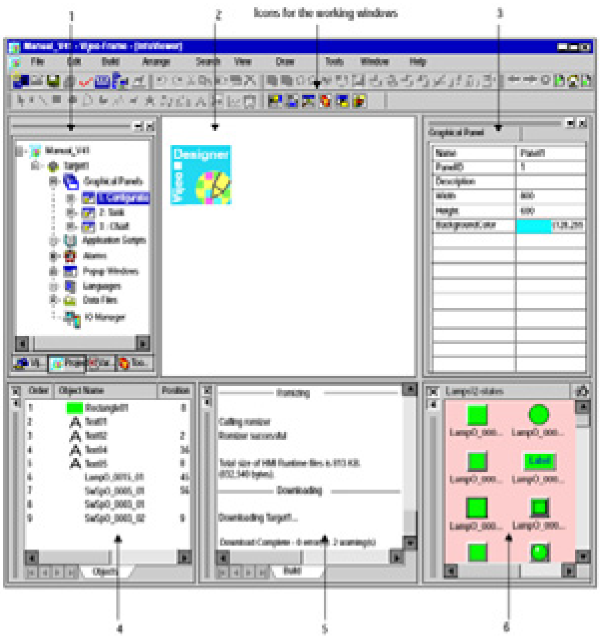
This application includes a Magelis XBT-GT 2330 type operator and display terminal, which communicates with the Controller Inside card and the Altivar 71 via the Modbus-RTU protocol.

The Vijeo Designer software is used to program and configure the terminal. The steps to be taken in order to create and download a program are described on the following pages.

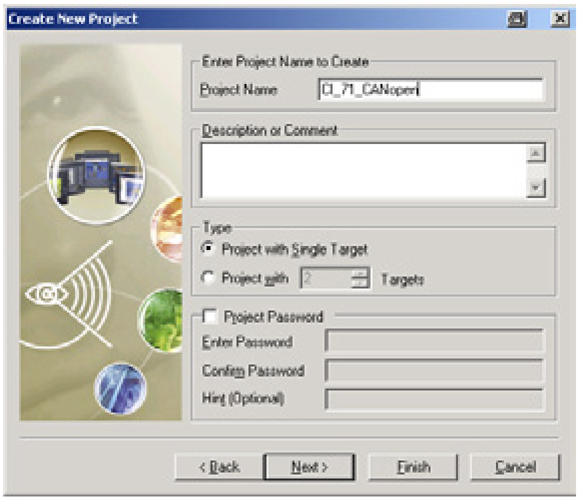
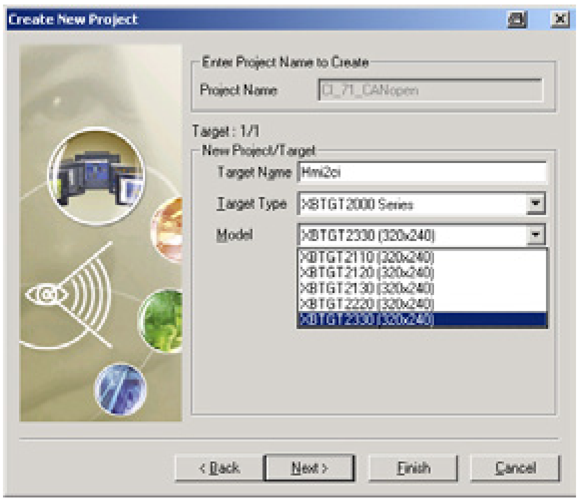
The HMI is set up as follows:

1. Vijeo Designer function overview
2. Create new project (specify platform, hardware, communication)
3. Communication settings
4. Set up new variables
5. Create screens
6. Show CANopen status
7. Check the project and download it
8. Application overview

Vijeo Designer Layout

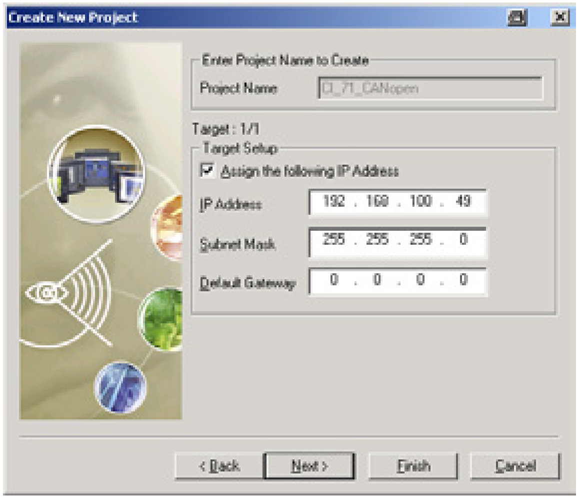
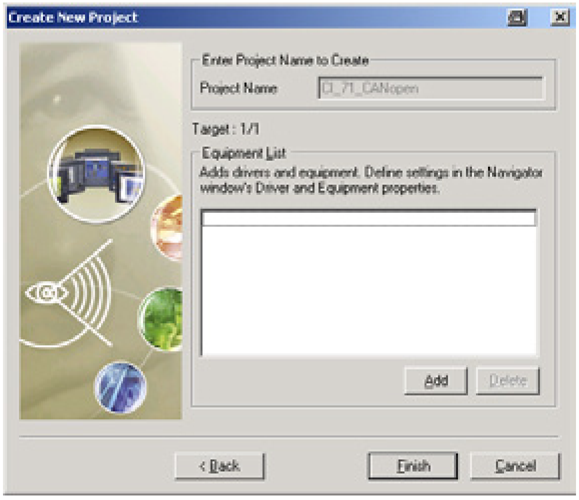
1	The Vijeo Designer environment consists of the following elements: 1 Navigator 2 Info-display 3 Inspector 4 Data list 5 Feedback-Zone 6 Toolbox	

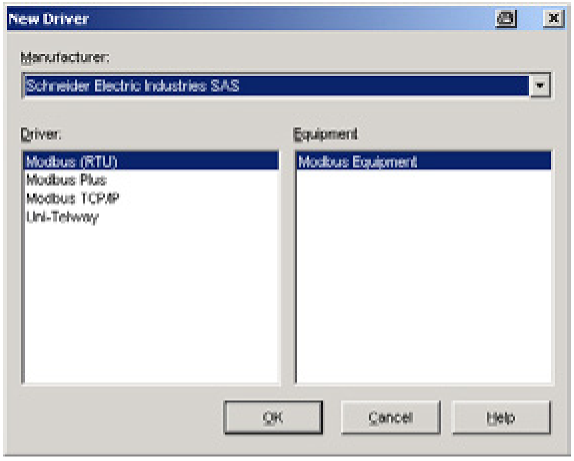
Creating a New Project

1	When Vijeo Designer starts up, a selection window appears. Select **Create new Project** and **Next.**	
2	Enter a **project name** for the application and a comment (if necessary).	
3	Next select the target device used and enter a logical name. Example project: • **Target Name: Hmi2ci** • **Target Type: XBTGT 2000 Series** • **XBTG Model: XBT-GT2330**	

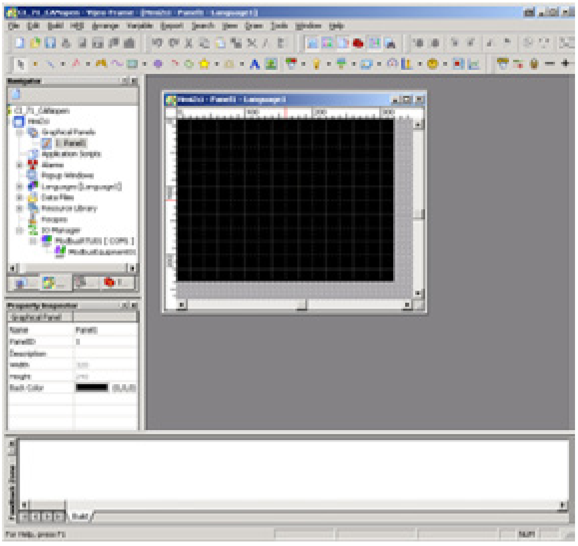
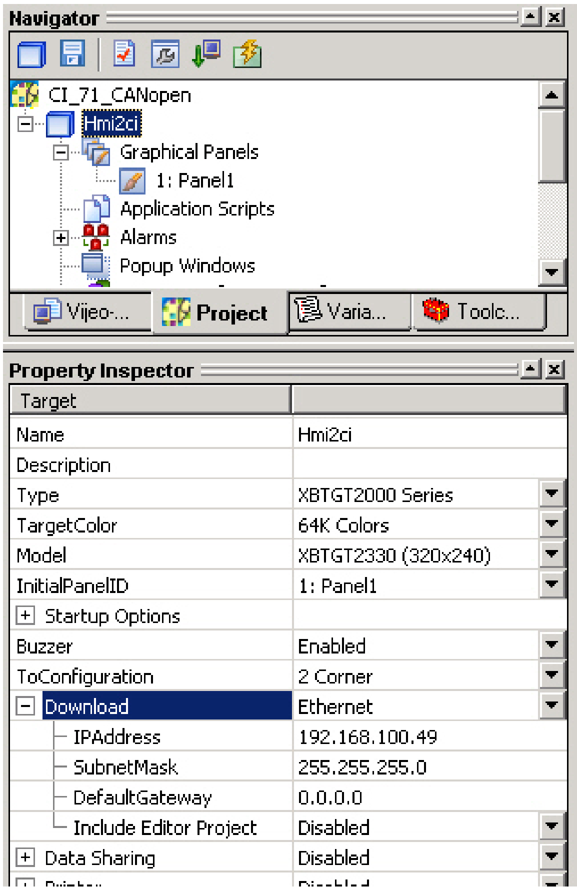
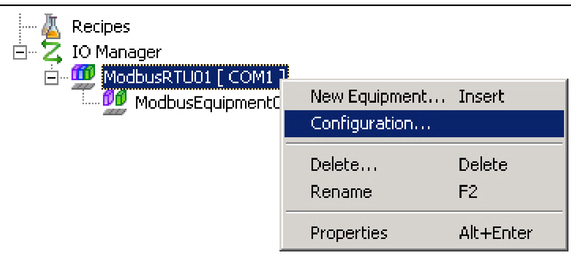
Continued on next page

Creating a New Project Contd.

<table>
<tr>
<td>4</td>
<td>In order to use the device's Ethernet interface, you need to enter the IP address, subnet mask and, if applicable, the gateway.</td>
<td>
</td>
</tr>
<tr>
<td>5</td>
<td>In order to be able to exchange data with other devices, the Magelis HMI requires a communication driver. To set one up, click the Add button.</td>
<td>
</td>
</tr>
<tr>
<td>6</td>
<td>Start by selecting Schneider Electric Industries SAS from the list under Manufacturer. Then select the Modbus (RTU) driver and Modbus Equipment (under Equipment) for communication with the Controller Inside PLC.

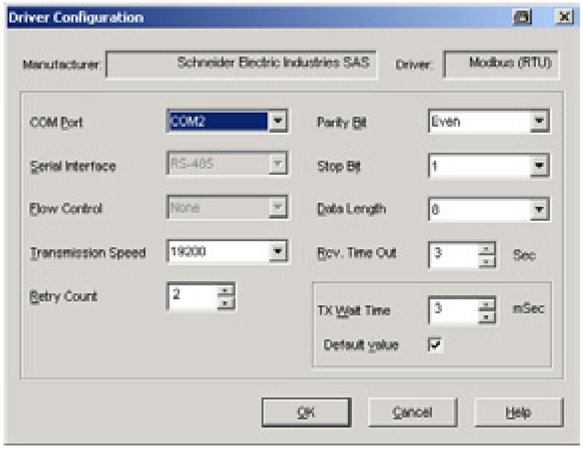
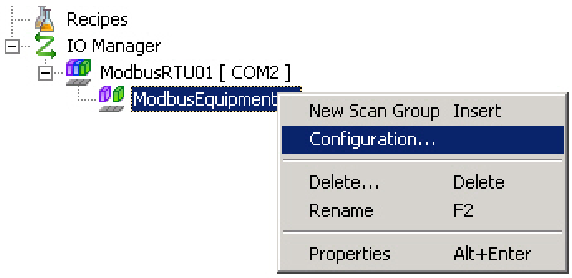
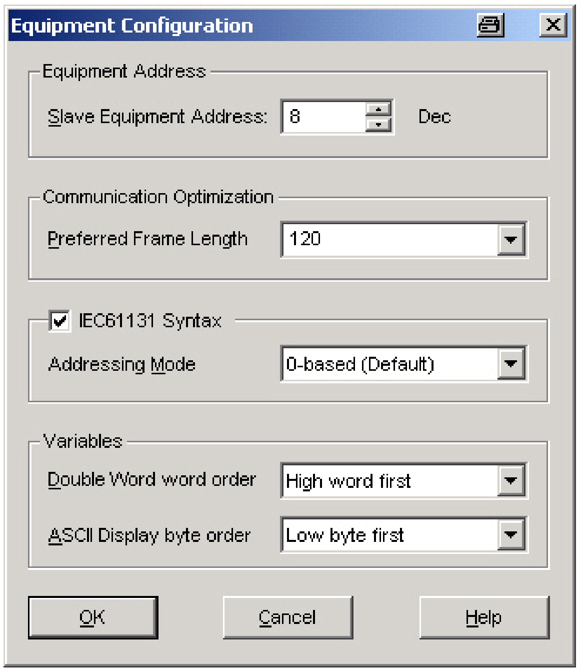
Once you have selected a communication driver, you can complete the creation of the new project by clicking the OK button followed by Finish.</td>
<td>
</td>
</tr>
</table>

Communication Settings

1	Once you have created the project, Vijeo Designer will display the workspace described above with an empty edit screen on the right-hand side.	
2	If the project is to be downloaded to the HMI via Ethernet, the settings can be modified here. To do this, right-click with the mouse on the target in the Navigator and select **Download** in the **Property Inspector**. In order that the project can be transferred to the Magelis HMI, you will need to select **Ethernet** as well as the **IP address** and the **subnet mask**.	
3	The interface parameters must be declared to the Modbus driver for communication with the PLC. Right-click with the mouse on **ModbusRTU01** and select **Configuration….**	

Continued on next page

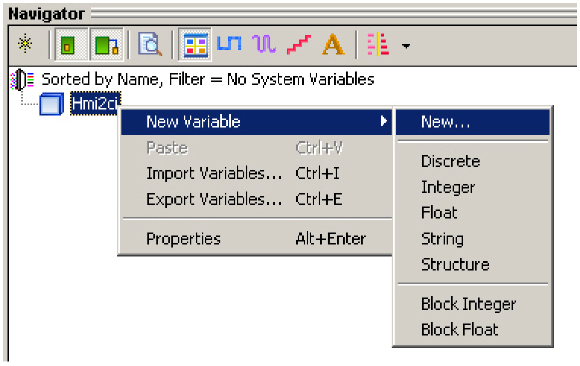
Communi-cation Settings Contd.

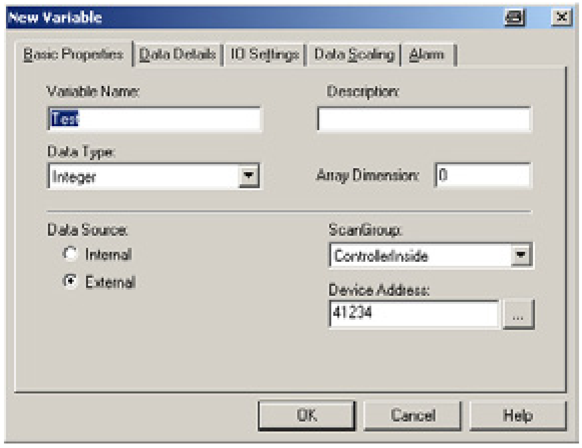
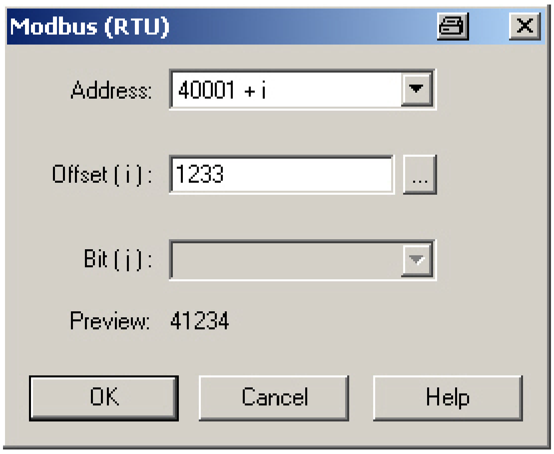
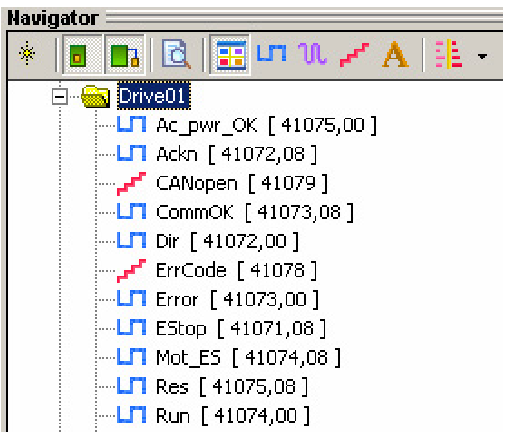
4	The setting entered here must be the same as in the PLC (Altivar 71). COM Port: **COM1** Serial interface: **RS485** Transmission Speed: **19200** **8 data bits, 1 stop bit, even**	
5	For the **equipment configuration**, right-click on **ModbusEquipment01** and select **Configuration….**	
6	Enter the **Modbus address** of the Controller Inside PLC.	
7	Right-click and select **Rename** to change the name.	

Continued on next page

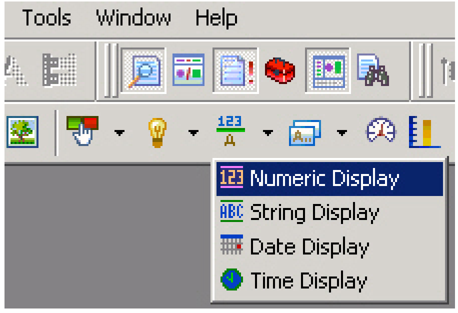
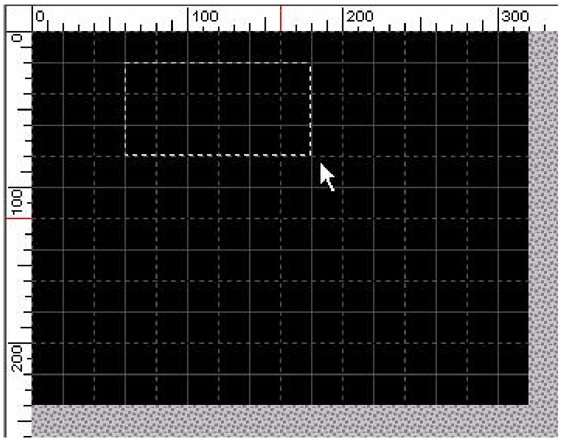
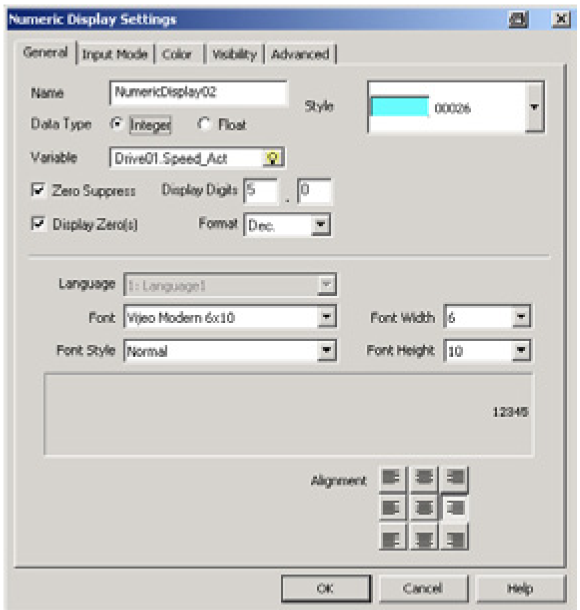
Creating Variables

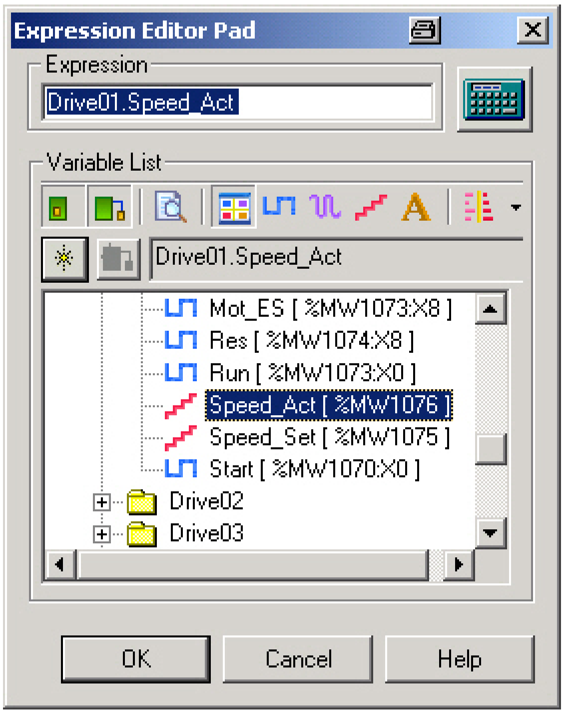
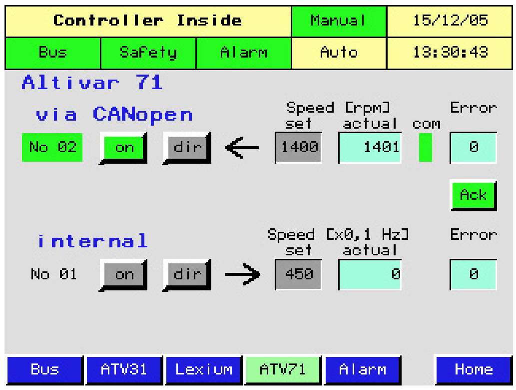
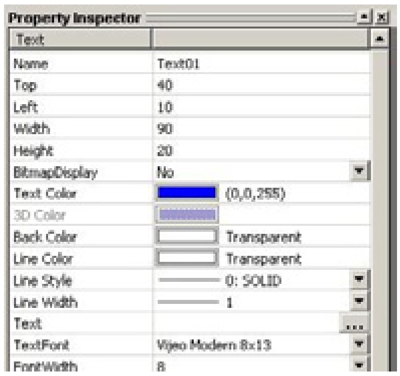
<table>
<tr><td>1</td><td>To create new variables in the Navigator, select the Variable tab at the bottom of the screen.

Right-click with the mouse on the project name to access a popup menu and select "New Variable → New .</td><td>
</td></tr>
<tr><td>2</td><td>To create variables, the following information must be entered:

• Variable Name
• Data Type
• Data Source (External)
• Address in the PLC</td><td>
</td></tr>
<tr><td>3</td><td>All PLC flags (in this case, 2000) can be addressed. Types that can be defined include flags (%M), words (%MW), double words (%MD) and floating points (%MF). All data to be displayed on the Viewer must be transferred to one of these types.</td><td>
</td></tr>
<tr><td>4</td><td>The variables created are displayed in the Navigator, along with their names and addresses.</td><td>
</td></tr>
</table>

Creating Screens

The process for creating animations on screens will now be described using a numerical example. The functions are similar for other animation elements.		
1	**Example: Insert Display** Selection from the menu bar. Various icons and elements are available in the menu bar and the toolbox.	
2	First, fix the size and position of the display on the panel.	
3	**Defining the Properties of the Display** You can define: • **Name** • **Data Type** • **Variable** • **Display style** • **Font style and size** The variable to be used for the animation can be input manually or selected using the bulb icon. If the varaible is undefined it is highlighted with red text.	

4	Zusätzliche Funktionen, z.B. die Invertierung des Wertes, können über das **Taschenrechner-Symbol** erzeugt werden.	
5	The display opposite shows the completed screen in which the separate properties for animation and actions appear.	
6	**Property Inspector** Each animation element on the screen has its own Property Inspector (right-click with the mouse) via which all settings associated with the element can be viewed and modified.	

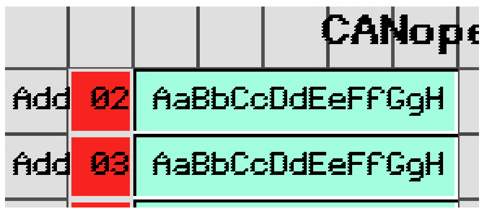
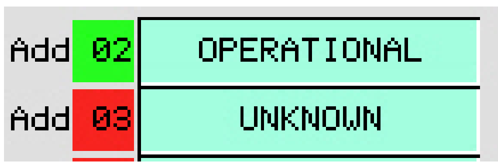
Showing CANopen Status

1	In the PLC, the status of each CANopen node is available as a number from 0 to 7. However, this needs to be displayed as text on the HMI. An application script is used for this purpose. Right-click to create a new script or change the name of a script.	

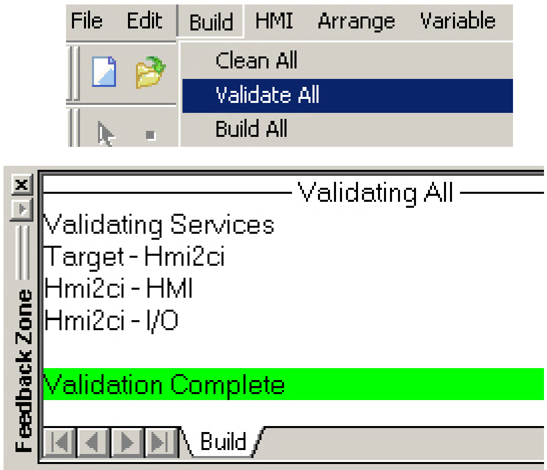
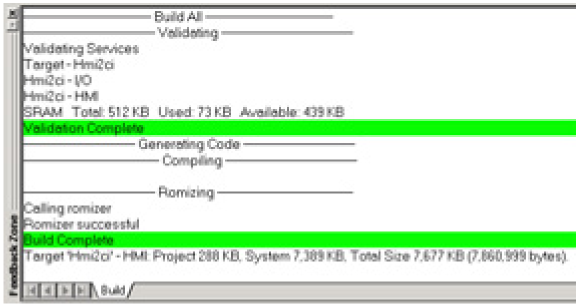
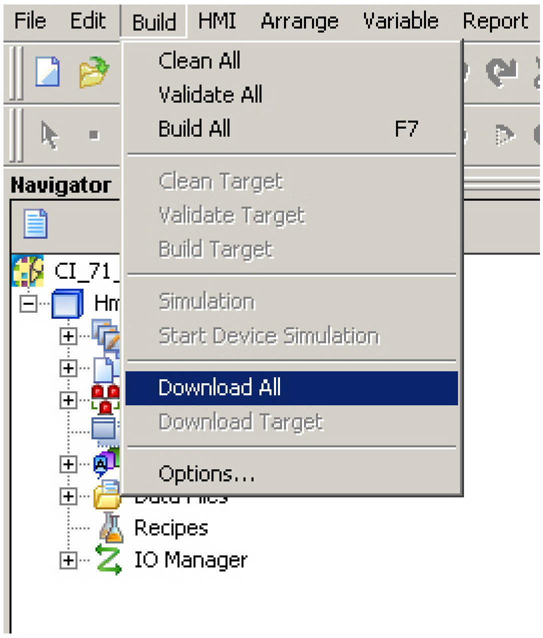
2	The process starts with the declaration of the variable and the reading of the CANopen status.	`int CANopen_no = 0;`
		`String CANopen_txt = "no input data";`
		
		`//--------------------------------------`
		`//      Copy input data for select text`
		
		`CANopen_no = Drive02.CANopen.getIntValue();`

3	Then the number is converted into the text you will see on the display.	`//--------------------------------------`
		`//   Select Text`
		
		`if (CANopen_no == 0)`
		`      { CANopen_txt = "INIT" ; }`
		`if (CANopen_no == 1)`
		`      { CANopen_txt = "RESET COMM" ; }`
		`if (CANopen_no == 2)`
		`      { CANopen_txt = "RESET APP" ; }`
		`if (CANopen_no == 3)`
		`      { CANopen_txt = "PRE OPERATIONAL" ; }`
		`if (CANopen_no == 4)`
		`      { CANopen_txt = "STOPPED" ; }`
		`if (CANopen_no == 5)`
		`      { CANopen_txt = "OPERATIONAL" ; }`
		`if (CANopen_no == 6)`
		`      { CANopen_txt = "UNKNOWN" ; }`
		`if (CANopen_no == 7)`
		`      { CANopen_txt = "NOT AVAIL" ; }`

4	Finally, the data is written to the output variable (type string).	`//-----------------------------------------`
		`//      Copy data for HMI`
		
		`CANopen_02.write (CANopen_txt);`

5	The item appears on the screen in text format (15 characters in length) (the entry is the above output variable).	
6	The output text will then appear during operation.	

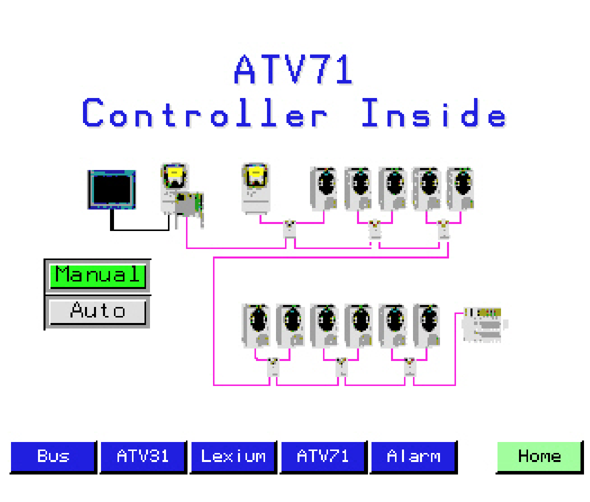
Downloading Project

1	Before being downloaded to the graphic HMI, the project must first be analyzed and validated. To do this, select: **Build->Validate All** The results are listed in the Feedback Zone.	
2	**Build All** performs a similar function.	
3	Select **Download All** under **Build** to transfer the application to the connected Magelis terminal. The configured communication route (Ethernet) will be used.	
4	**Defining the Ethernet IP Address** If you have chosen the Ethernet connection to download your project and you have never loaded a project into the HMI before, you must define the IP address of the HMI. To do this, touch the screen of the HMI in the top left corner whilst turning it on. This will start the HMI **Runtime** and allow you to set up the address in the **Offline** tab.	

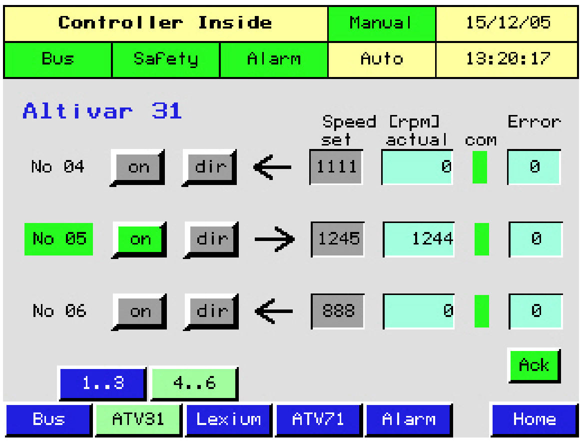
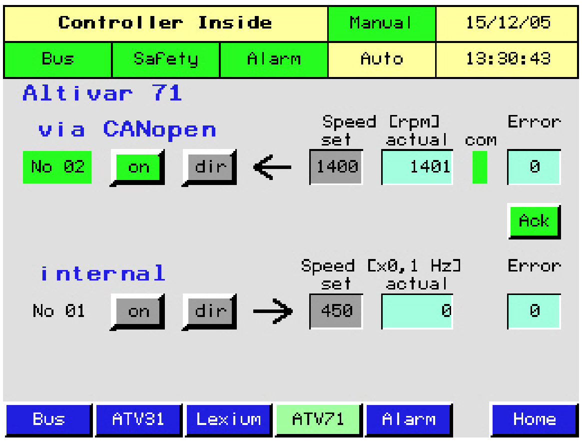
Application Overview

<table>
<tr><td>1</td><td>The example application features a number of displays that can be selected by the user.

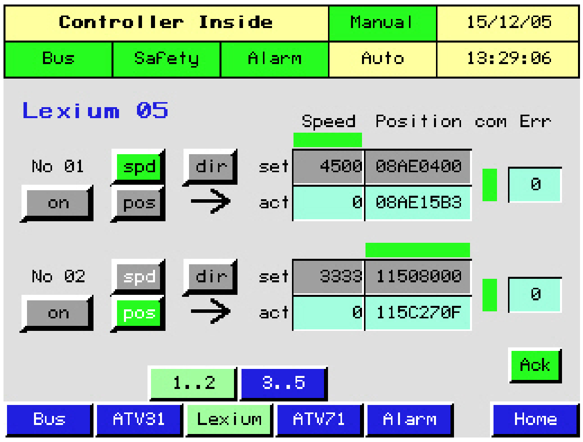
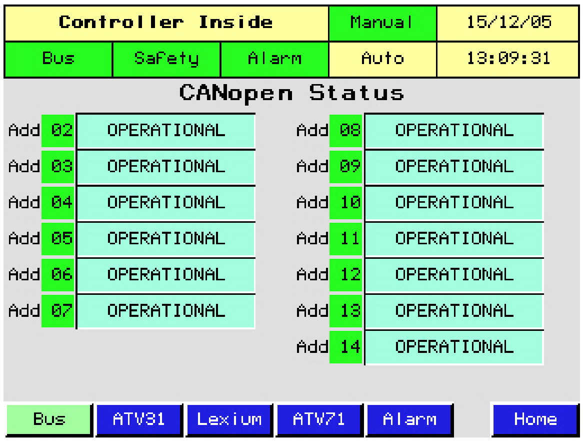
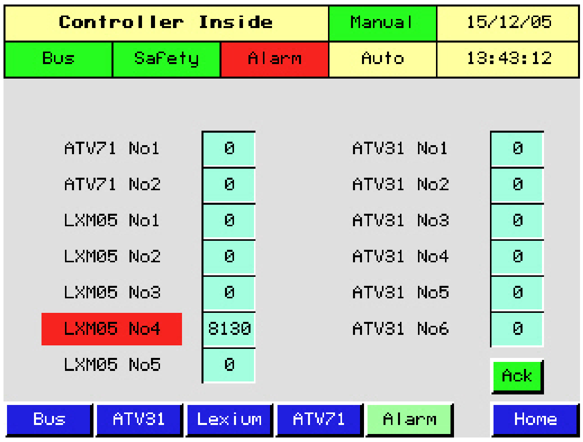
The structure is mapped on the welcome screen. This is also where the operating mode can be selected. There are no logic configuration settings in the PLC for automatic mode.

All drives can run in manual mode, controlled directly via the Viewer. To do this, you must switch to the relevant screen.</td><td>
</td></tr>
<tr><td>2</td><td>The screen opposite shows three Altivar 31 drives. For each one, there is a button for starting/stopping and selecting the direction of rotation. It is also possible to select the setpoint speed for the drives in the Set field.

The status message and actual speed display, along with the error code, act as feedback.

The header on subsequent screens is identical and provides information about the status of the machine.</td><td>
</td></tr>
<tr><td>3</td><td>ATV71 drives are controlled in a similar way to ATV31 drives.</td><td>
</td></tr>
</table>

Application Overview Contd.

4	The corresponding screen for the first two Lexium 05 servo drives appears opposite. Two modes (speed and position) are available for selection.	
5	If a CANopen bus node is faulty, this will be indicated in the header (Bus button). Go to the bus screen to identify the node.	
6	The same applies in the event of an error message pending for a drive. General message via header. Detailed information on alarm screen.	

PLC

Introduction

The PLC chapter describes the steps required for the initialization and configuration and the source program required to fulfill the functions.

Pre-conditions

Before carrying out the steps described below, you must ensure that:

- The CoDeSys PS1131 programming tool is installed on your PC
- The TemplateProject.pro PLC project is available in the default directory that has been set up (C:\Program Files\Schneider Electric\PS1131\CoDeSys V2.3\Targets\ControllerInside\Examples\)
- The Controller Inside card is connected to the power supply
- The PLC and the PC are linked to one another via the PC <> Controller Inside programming cable (VW3A8106)

To simplify programming, we are going to use the TemplateProject.pro startup project recommended and supplied with the PS1131 tool. It already contains the basic functions that might need to be enabled or extended. Reconfiguration is beyond the scope of this description.

Configuration

Setting up the PLC is done as follows:

<u>General</u>

- Create new program
- Add program setting and LED control to recommended startup project template.pro
- Download program to PLC and start up
- Create data structure
- Create variables

<u>CANopen</u>

- Link CANopen master
- Integrate CANopen EDS files
- Link Altivar 71 for CANopen
- CANopen expansion in main program
- Create function block (ST)
- Create function block (ST) (example ATV71)
- Create program block (FBD) for ATV71
- Link Lexium05 for CANopen
- Special features in function block (ST) for LXM05
- Create program block (FBD) for LXM05
- Link Altivar 31 for CANopen
- Special features in function block (ST) for ATV31
- Create program block (FBD) for ATV31
- Link Advantys STB I/O island for CANopen

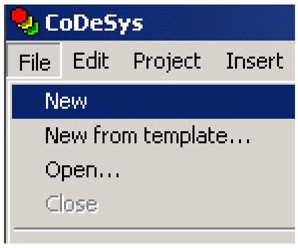
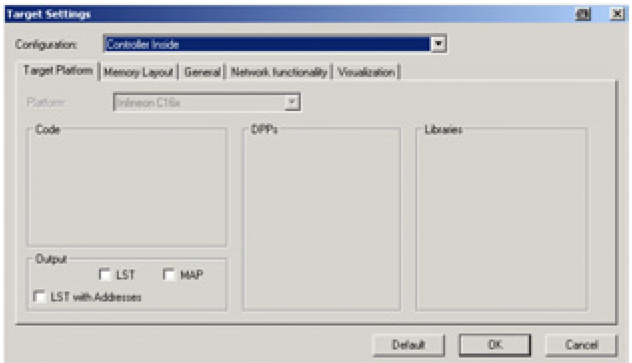
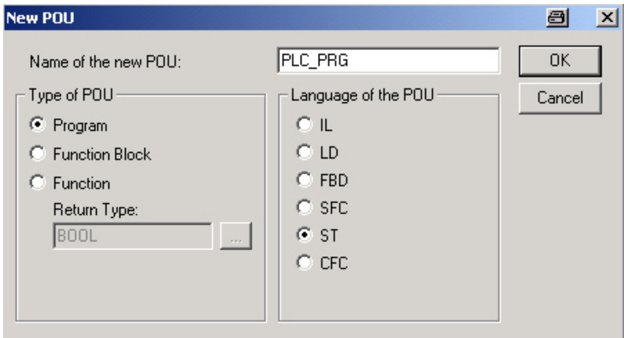
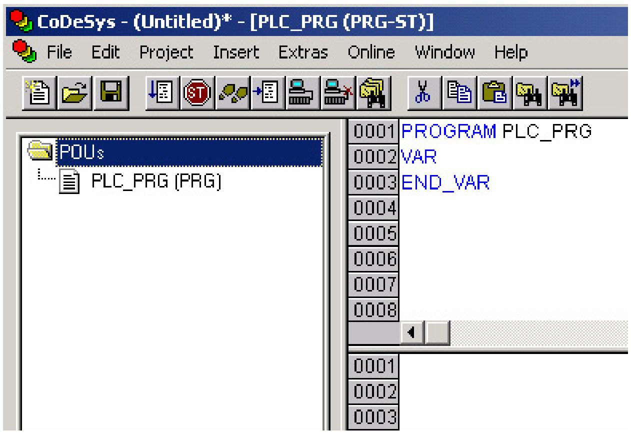
<u>Internal communication</u>

- Altivar 71 for internal data exchange
- Use plug-in graphic display terminal
- Create viewer within CoDeSys

<u>External HMI</u>

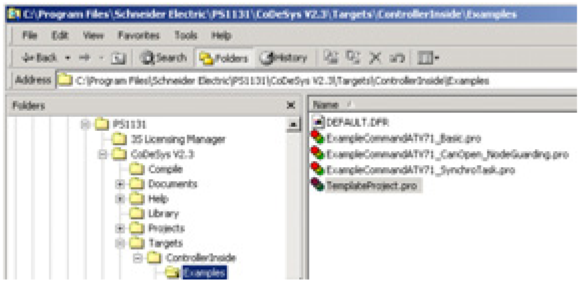
- Data exchange with an external HMI

Creating a New Program

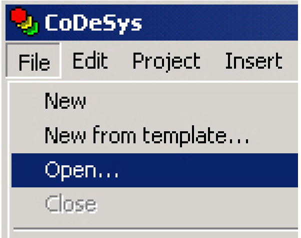
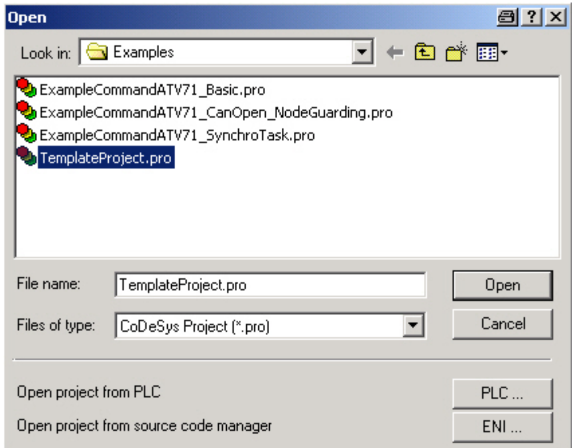
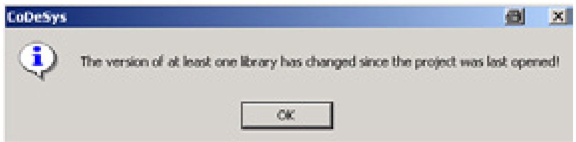
1	To create a new application, once the PS1131-CoDeSys software has started up, select: **File->New**	
2	Select **Controller Inside** as the target system in the next window that appears.	
3	You can make further settings here. We are going to accept the default settings.	
4	Accept the proposed start block. **Note:** Do not delete or rename the PLC_PRG program block unless you are not using a task configuration. PLC_PRG is usually the main program in a single-task program.	
5	Confirm your settings to complete the programming setup for PS1131-CoDeSys. You now need to set up various general programs. Reconfiguration is beyond the scope of this description. We are, therefore, going to use the startup project TemplateProject.pro recommended and supplied with PS1131-CoDeSys.	

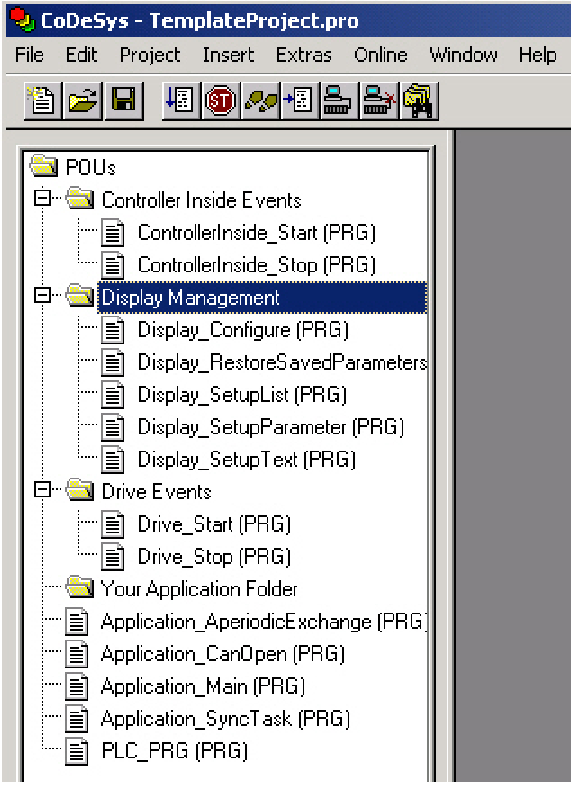
Expanding the Recommended Startup Project TemplateProject.pro

<table>
<tr><td>1</td><td>Following installation of PS1131-CoDeSys, the startup project will be available under:

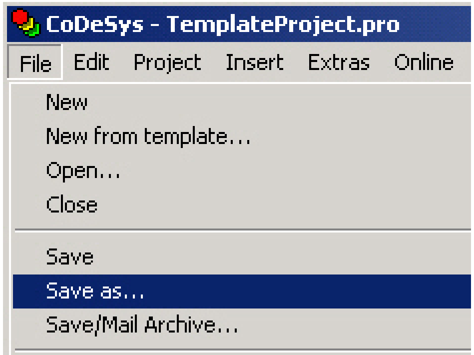
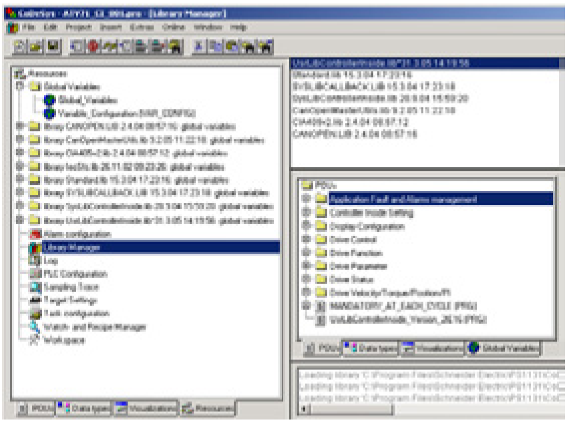
<Installation path>\Targets \ControllerInside\Examples \TemplateProject.pro</td><td>
</td></tr>
<tr><td>2</td><td>Once the PS1131-CoDeSys software has started up, select:

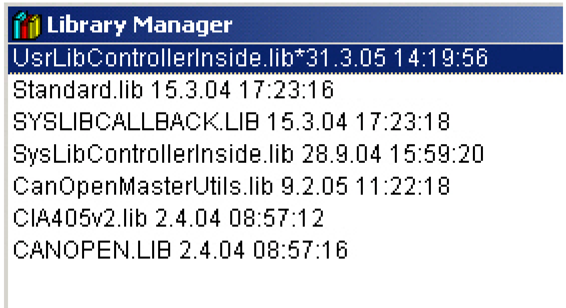
File->Open….

……</td><td>
</td></tr>
<tr><td>3</td><td>…to select and confirm the startup project.</td><td>
</td></tr>
<tr><td>4</td><td>If the version of the library has changed, a corresponding message will appear on the screen. Click OK to confirm.</td><td>
</td></tr>
<tr><td>5</td><td>The project is write-protected. However, you can save it under a new name.</td><td>
</td></tr>
</table>

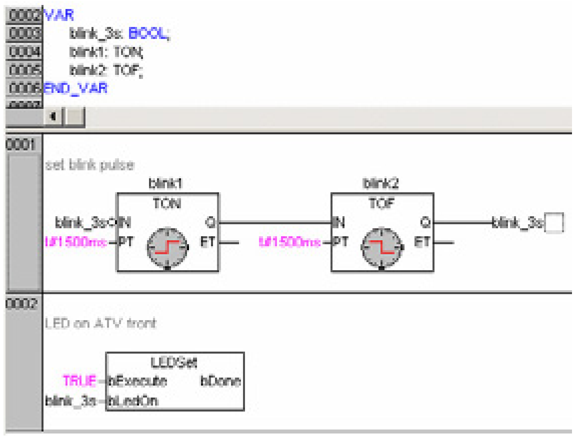
<table>
<tr><td>6</td><td>Once you have clicked OK, the startup project will open and appear on the screen.</td><td>
</td></tr>
<tr><td>7</td><td>Select

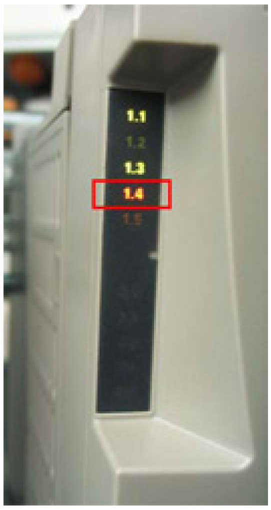
File->Save as..

to save the project with a new name.</td><td>
</td></tr>
<tr><td>8</td><td>Select Library Manager on the Resources tab in the Object Organizer.</td><td></td></tr>
</table>

<table>
<tr>
<td>9</td>
<td>The libraries listed opposite are integrated in PS1131-CoDeSys.</td>
<td>
</td>
</tr>
<tr>
<td>10</td>
<td>This is a simple program to make LED 1.4 on the front panel of the ATV71 flash.

Different LED flashing frequencies are possible to indicate different states.

This setting is made in the Application_Common program, which is called from the Application_Main program.</td>
<td>

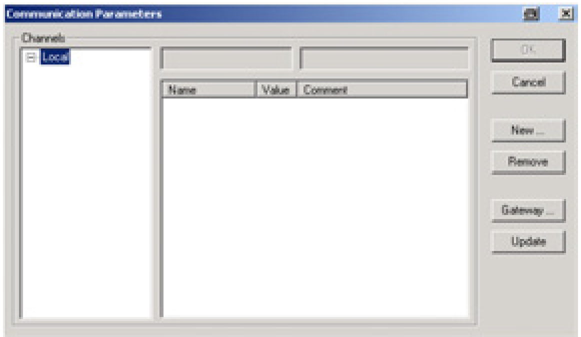

</td>
</tr>
</table>

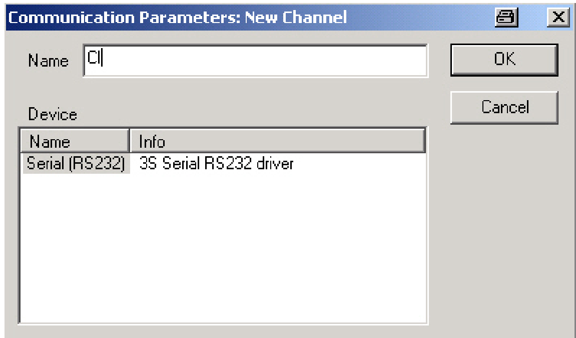
Download & Start Up

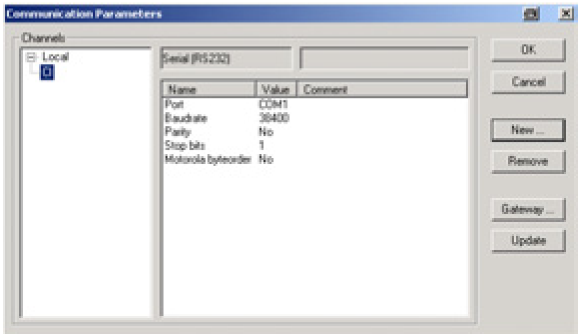
<table>
<tr><td>1</td><td>For the purpose of downloading, the PC must be connected to the Controller Inside card. Connection cable VW3A8106 is used for this purpose.

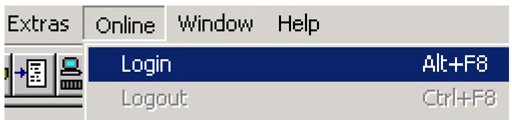
Select:
Online->Communication Parameters

in the menu bar.

The current data appears. If no data appears, you can add a new channel by clicking New.</td><td>
</td></tr>
<tr><td>2</td><td>You can change the name of the new channel.

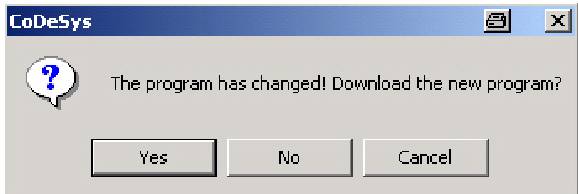
Click OK to confirm.</td><td>
</td></tr>
<tr><td>3</td><td>The new channel appears.

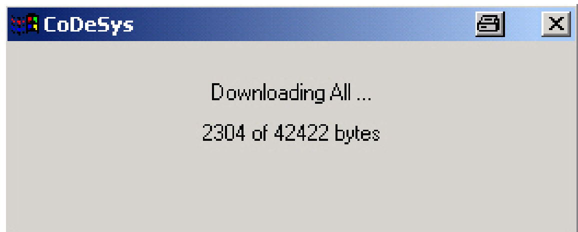
Click OK to close the window.</td><td>
</td></tr>
<tr><td>4</td><td>The connection with the PLC is established by selecting:

Online -> Login.</td><td>
</td></tr>
<tr><td>5</td><td>The project must be free from errors.

When the connection is established, the software will check that the programs on the PC and PLC are identical.

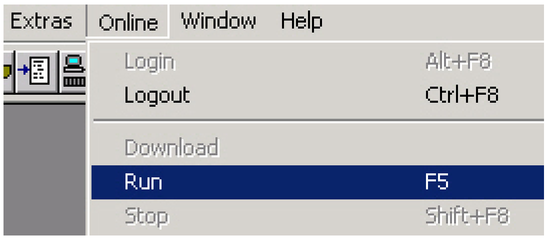
If they are not, it will prompt you to download the program to the PLC.

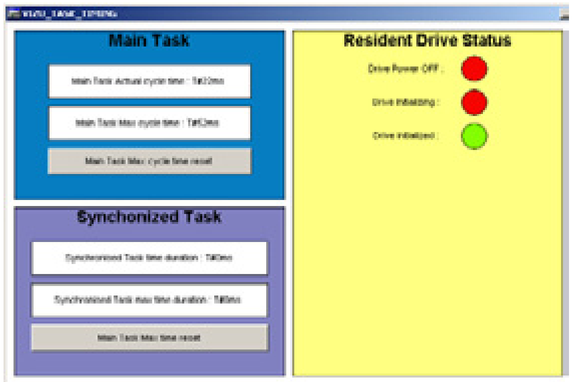
Select Yes to start the down-load.</td><td>


</td></tr>
</table>

<table>
<tr><td>6</td><td>Once the download is complete, the program must be started.

Do this by selecting

Online -> Run.</td><td>
</td></tr>
<tr><td>7</td><td>Information appears in the status bar in the bottom right-hand corner of PS1131.</td><td>
</td></tr>
<tr><td>8</td><td>You can also view the state of the Controller Inside card in the preconfigured PS1131-CoDeSys viewer.

To do this, open the VIZU_TASK_TIMING window via the Visualizations tab in the Object Organizer.</td><td>
</td></tr>
</table>

Create Boot-Project

<table>
<tr><td>1</td><td>Use the function

Create boot project

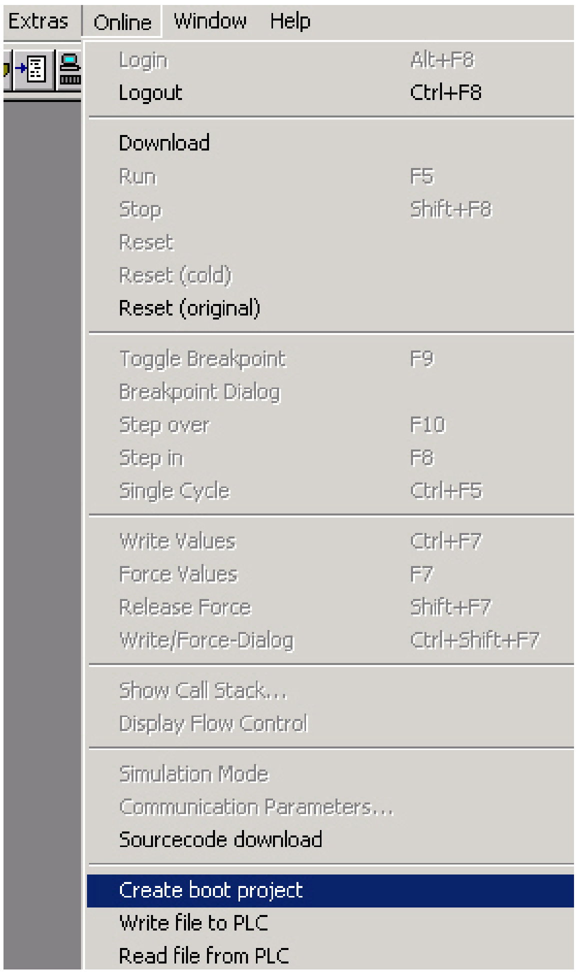
To copy the program into flash memory so that the PLC automatically loads the project on start-up.

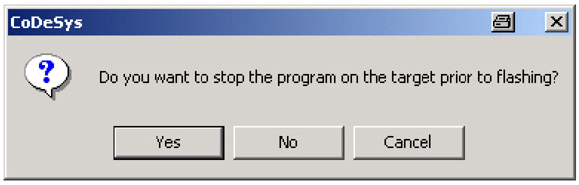
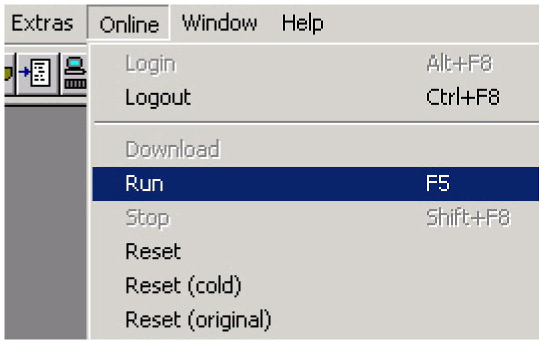
.

Note:
Under Project->Options in the category Source download:

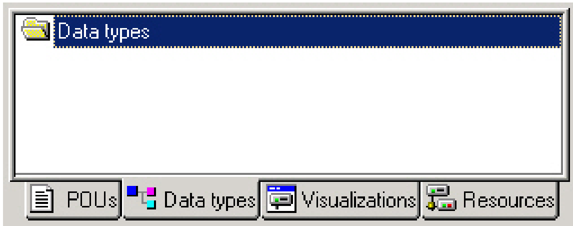
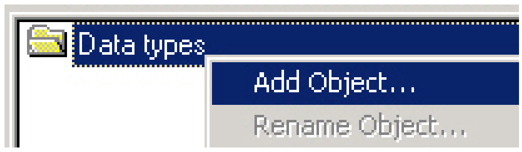
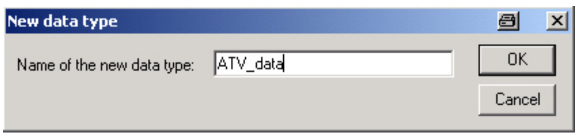
If you activate the radio button

Implicit on create boot project

The project is automatically downloaded to the PLC when the function Create boot project is finished.</td><td>
</td></tr>
</table>

2	The PLC must be stopped in order to perform the download. Select **Yes**.	
3	A message informs you that the project is loading	
4	When finished (and if required) you must start the PLC again with **Online->Run**	

Creating the Data Structure

1	Select the **Data types** tab in the Object Organizer. Data structures are used to group common data types.	
2	Click with the right mouse button and select: **Add Object….**	
3	Give the data type a name.	
4	The **ATV_data** data structure is used to link the Altivar function blocks to the I/O communi-cation addresses or data.	TYPE ATV_data : STRUCT data: WORD; (* DriveCom data, IN=status, OUT=command*) speed: WORD; (* speed value, IN=actual, OUT=setpoint *) error: WORD; (* error code, IN=actual, OUT= not used *) END_STRUCT END_TYPE
5	The following data structures are used: • ATV_data I/O data for Altivar FBs • ATV_HMI Data exchange with HMI for Altivar • LXM_data I/O data for Lexium FBs • LXM_HMI Data exchange with HMI for Lexium	

Creating Variables

Variables are declared either in locally in the declaration section of a block or in global variable lists.

Note:
A local variable can be defined with the same name as a global variable. Within a block, the variable defined locally always takes priority. You cannot give two globally defined variables the same name (for example, a compilation error will occur if a variable called "var1" both appears in a global variable list and has been declared in the control configuration).

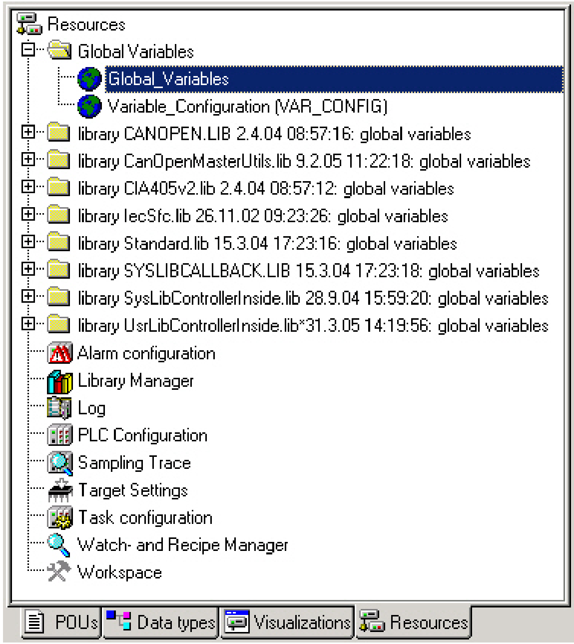
In respect of the names of variables, do not include blank spaces or (in German) umlauts. Variables can only be declared once and must not be identical with keywords. The names of variables are not case-sensitive (i.e. VAR1, Var1 and var1 will be one and the same variable). In names, underscores are significant, e.g., "A_BCD" and "AB_CD" will be two different variables. Do not use more than one underscore in succession at the start of a name or within a name. Variables can be used wherever permitted by the declared type.

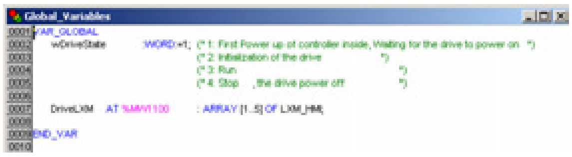
You can call up a list of available variables via the input assistant.

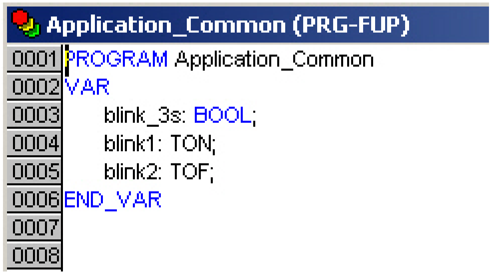
1

- "Normal" variables
- Constants or
- Remanent variables

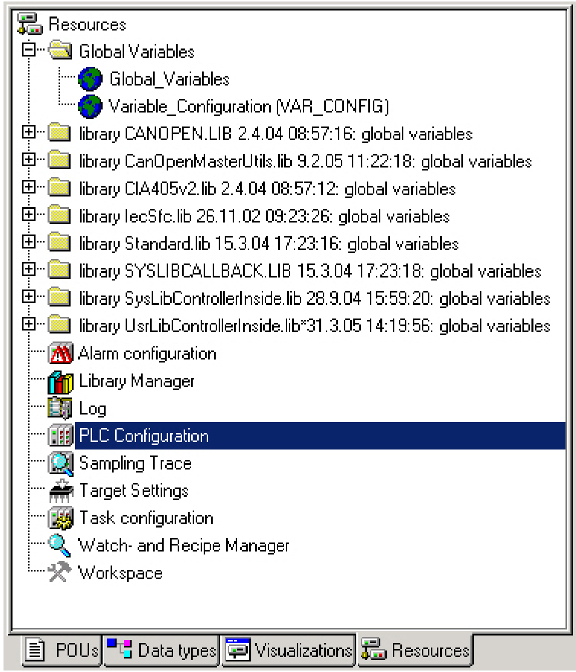
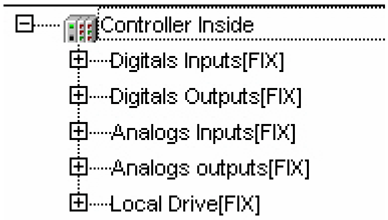
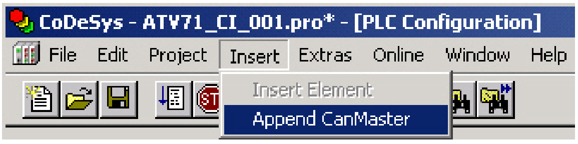
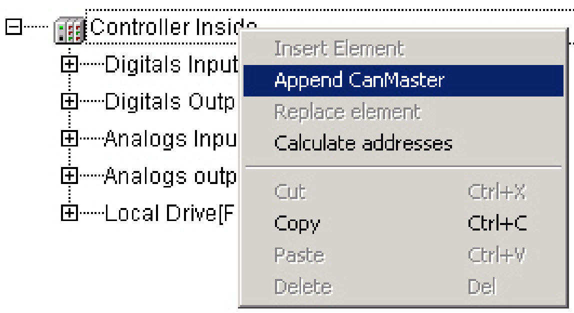
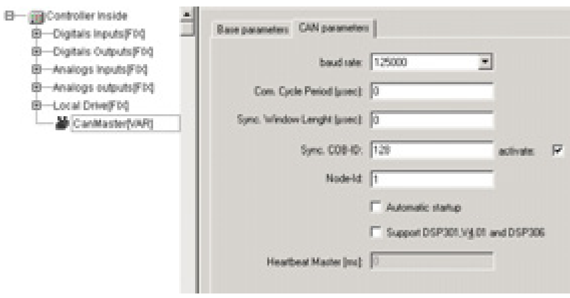
defined in the overall project can be declared as global variables.





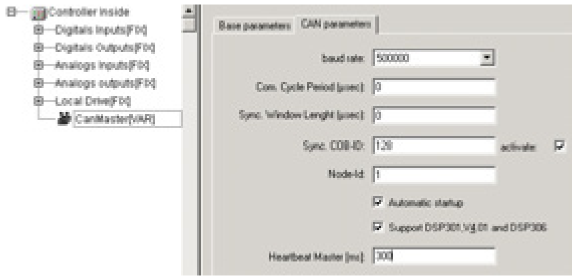
2	In the declaration section of a block, all variables used only in that block are declared. These might be: • Input or output variables • I/O variables • Local variables • Remanent variables and • Constants The declaration syntax is based on the IEC 61131-3 standard. Please note that it is possible to use object templates during the initial stages of declaration when creating a new ‘Global variables’, ‘File type’, ‘Function’, ‘Function block’ or ‘Program’ object.	

Linking a CANopen Master

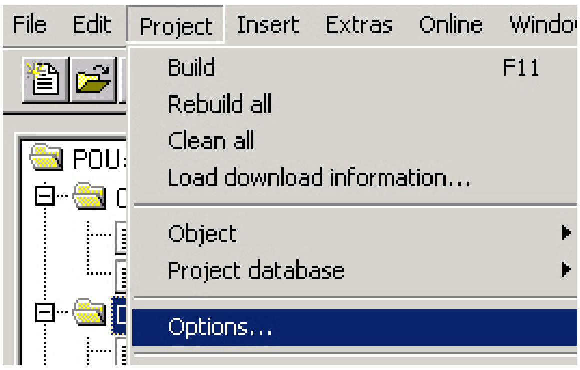
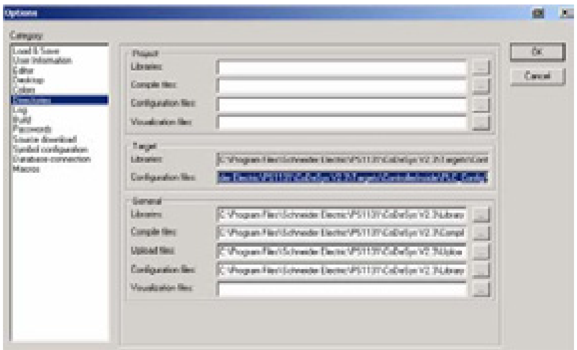
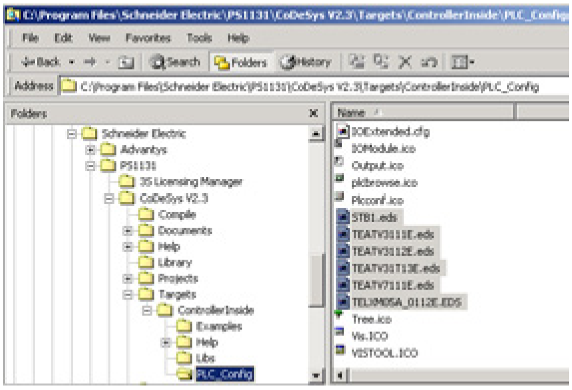
1	Select **PLC Configuration** on the **Resources** tab in the Object Organizer.	
2	The PLC configuration appears on the right-hand side of the window.	
3	Select: **Insert->Append CanMaster** to link a CANopen master or right-click on Controller Inside.	
4	When you select **CanMaster**, a properties dialog box will appear on the right-hand side. You can accept the **Base parameter** default settings. The **CAN parameters** tab displays the parameters.	

<table>
<tr>
<td>5</td>
<td>It is here that global settings and monitoring parameters for the CANopen bus are defined. Select the required baud rate for transmission on the bus. We are using the following settings for this project:

• Baud rate: 500,000 bps
• Automatic startup
• Support DSP …

The heartbeat is set in the Application_CanOpen program.</td>
<td>
</td>
</tr>
</table>

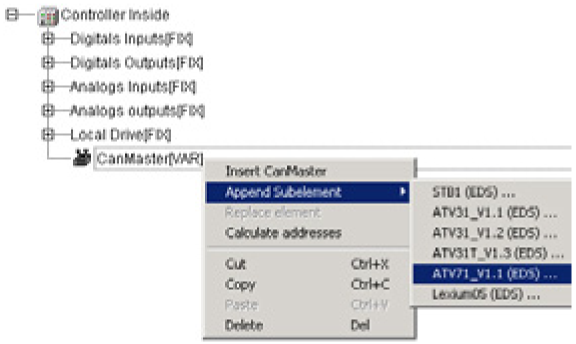
Integrating the CANopen EDS Files

1	In order that devices (subelements) can be appended to the CANopen master, they must be declared in PS1131 (CoDeSys). The device-specific EDS files are used for this purpose. Select **Project -> Options** to open the Options dialog box and check the target directory.	
2	The target directory appears in the **Directories** category under **Target** and **Configuration files**. If you did not change any of the default settings during installation, the target directory will be: C:\Program Files \Schneider Electric\PS1131 \CoDeSys V2.3\Targets \ControllerInside\PLC_Config\	
3	Copy the EDS files for • Altivar 71 • Altivar 31 • Lexium 05 and • Advantys STB to this directory. **Note:** The EDS files for the VSDs and servos appear on the CDs supplied with the relevant products. For Advantys STB, these files are created with the Advantys Configuration Software. You will need to **restart** PS1131 (CoDeSys) once the process is complete!	

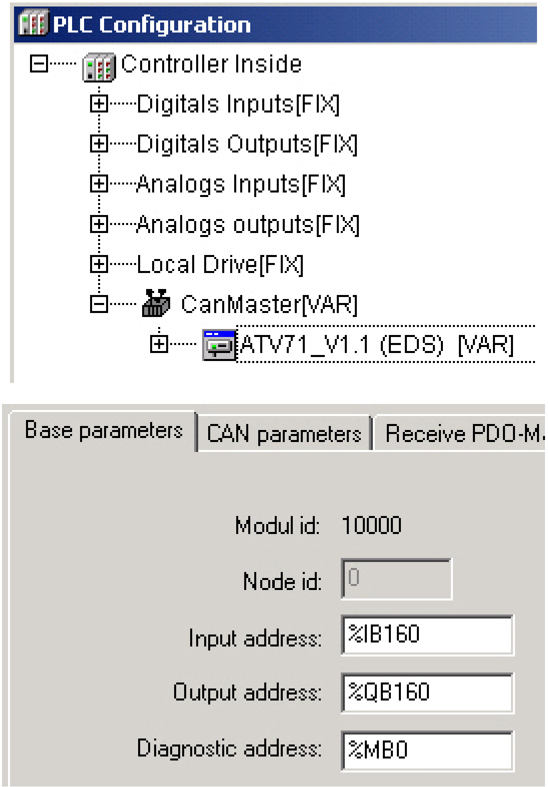
Linking Altivar 71 for CANopen

<table>
<tr>
<td>1</td>
<td>Once you have copied the EDS file, right-click with the mouse on the CanMaster and select the ATV71 subelement.

This function can also be selected in the menu under

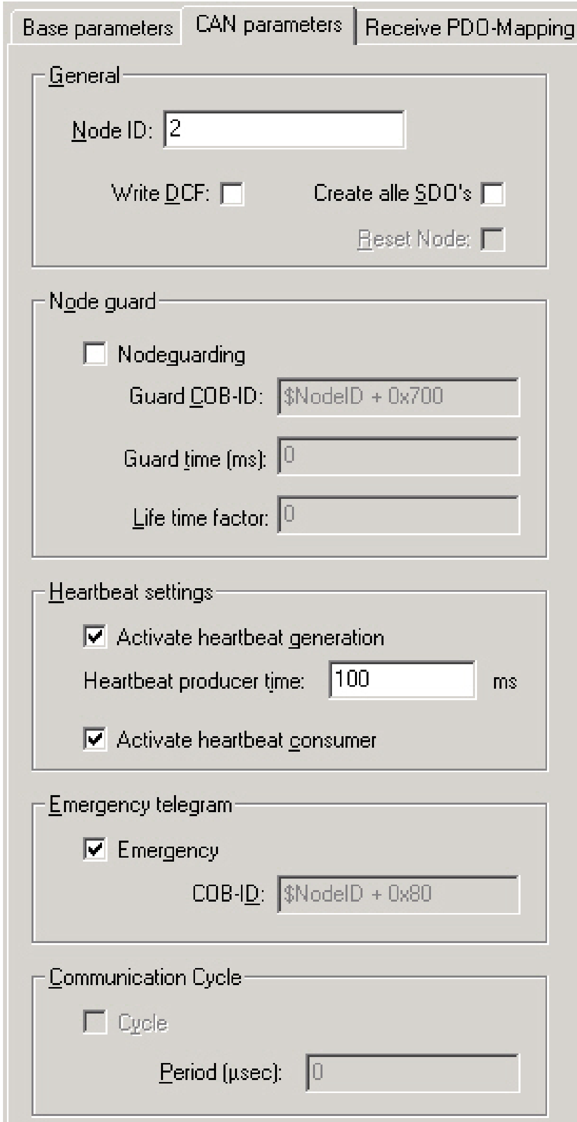
Insert->Append Subelement.</td>
<td>
</td>
</tr>
<tr>
<td>2</td>
<td>Once you have selected the ATV71, a properties dialog box will appear on the right-hand side.

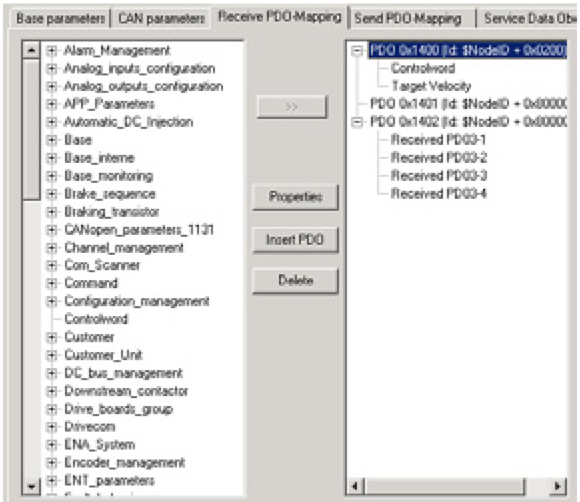
On the Base parameters tab, enter %IB160 as the input address and %QB160 as the output address.
(See the list in the Communication chapter for more information.)

Note:
A start address of %QW80 is the same as one of %QB160 and %QD40.</td>
<td>
</td>
</tr>
</table>

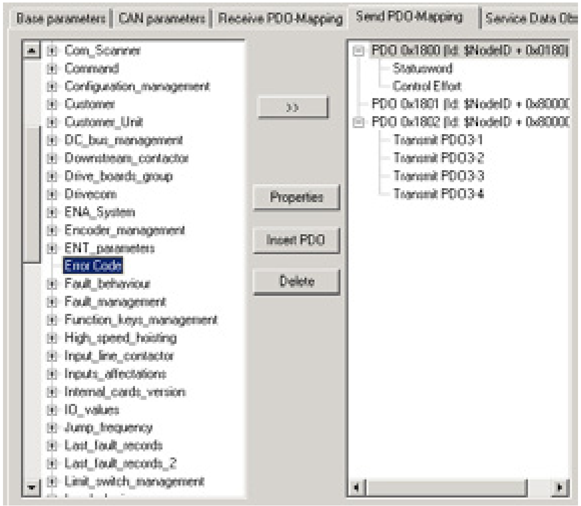
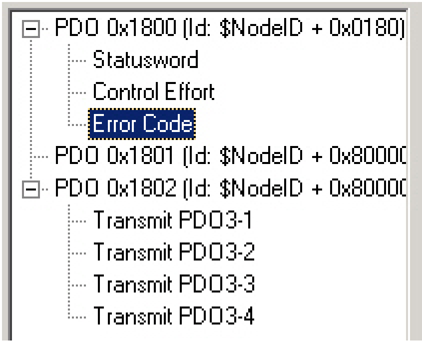
<table>
<tr>
<td>3</td>
<td>The CANopen address 2 is entered in the Node ID field on the CAN parameters tab.

Nodeguarding is unchecked; Heartbeat and Emergency are checked.

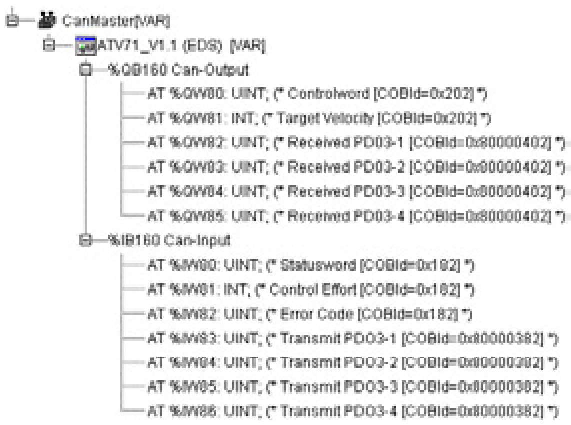
The Heartbeat producer time is 100 ms.</td>
<td>
</td>
</tr>
<tr>
<td>4</td>
<td>On the Receive PDO-Mapping tab (Drive <- PLC), the default setting for the first PDO is accepted.

The second and third PDOs remain deactivated.</td>
<td>
</td>
</tr>
</table>

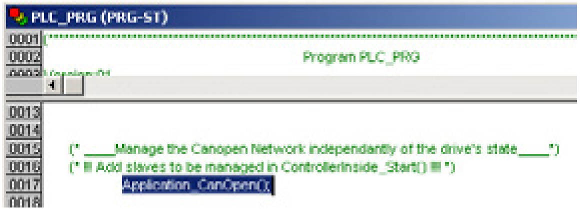
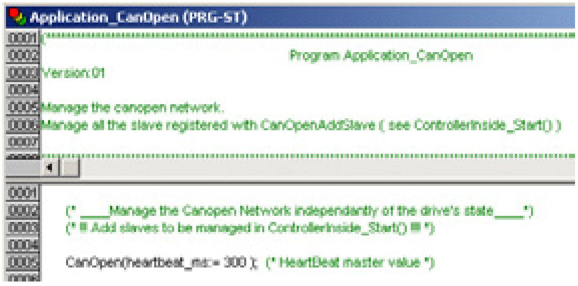
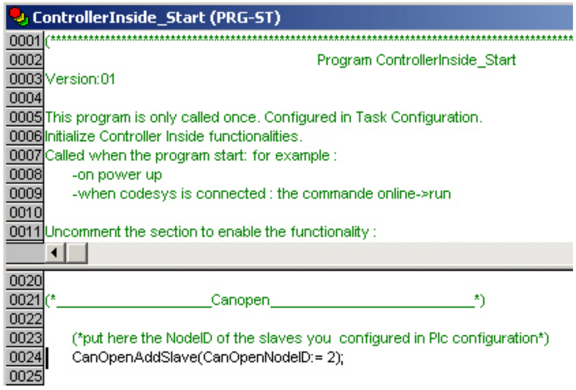
<table>
<tr><td>5</td><td>On the Send PDO-Mapping tab (drive -> PLC), the Error Code entry is added to the default setting for the first PDO.

This is done by selecting the Error Code entry in the left-hand column and clicking >> to add it to the right-hand column. The second and third PDOs remain deactivated.</td><td>
</td></tr>
<tr><td>6</td><td>Once added, the Error Code appears as follows.</td><td>
</td></tr>
<tr><td>7</td><td>The address assignment appears in the PLC configuration.

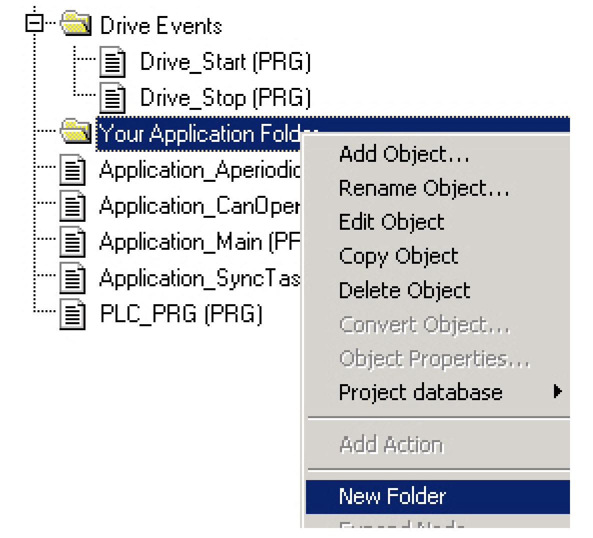
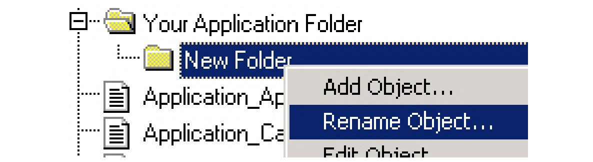
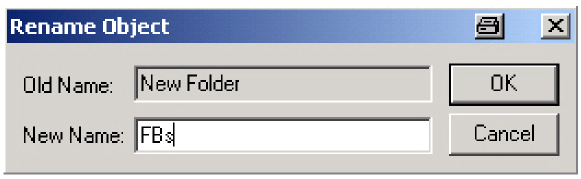
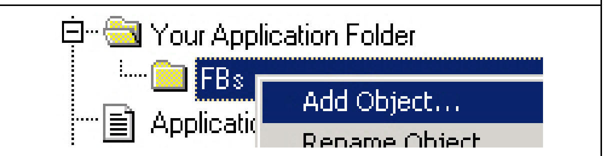
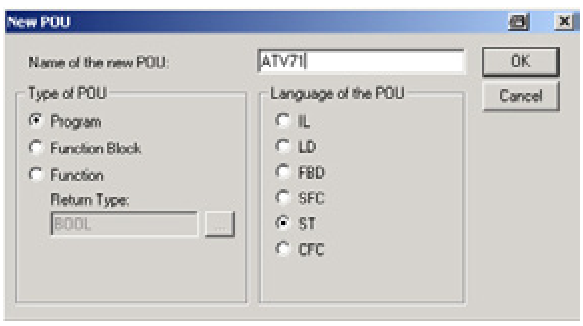
%QW80 = Control word
%QW81 = Setpoint speed
%IW80 = Status word
%IW81 = Actual speed
%IW82 = Error word

The following addresses are reserved for the drive but are not used (PDOs are deactivated):
%QW82 ... %QW85 and
%IW83 ... %IW86</td><td>
</td></tr>
</table>

CANopen Expansion in the Main Program

1	The **startup project** makes provision for the integration of CANopen. Select: **Application_CanOpen();** to call up the **PLC_PRG** program.	
2	Call up the **CANopen block** in the **Application_CanOpen** program. CanOpen(heartbeat_ms:=300)	
3	When the PLC starts up, the **ControllerInside_Start** program is called once and the connected CANopen nodes are declared. The node with address **2** is added here. **CanOpenAddSlave(CanOpen NodeID:= 2);** Subsequently, all node addresses (2…14) will be entered here.	

Creating a function block

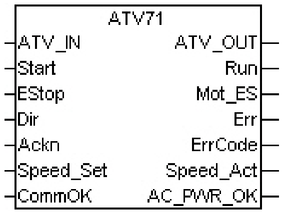
1	**Function block** Select the **Blocks** tab in the Object Organizer. To ensure clarity, a separate folder is created for the FBs. Click with the right mouse button on the existing folder and select **New Folder**.	
2	Click with the right mouse button again to **rename** the new folder.	
3	In this case we are going to use the **FBs** subdirectory.	
4	To add a function block, click with the right mouse button and select **Add Object...**.	
5	Select **Function Block** under **Type of POU** in the next window. You also need to define the POU name and language. In this case, these are **ATV71** and **ST** respectively.	
6	The following function blocks are used: • ATV71 FB to control Altivar 71 • ATV31 FB to control Altivar 31 • LXM05 FB to control Lexium 05 • from_ATV Status/actual value from Altivar to FB • to_ATV Commands/setpoints to Altivar • from_LXM Status/actual value from Lexium 05 to FB • to_LXM Commands/setpoint to Lexium 05	

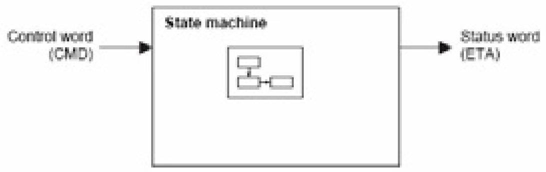
Creating a function block (ST)

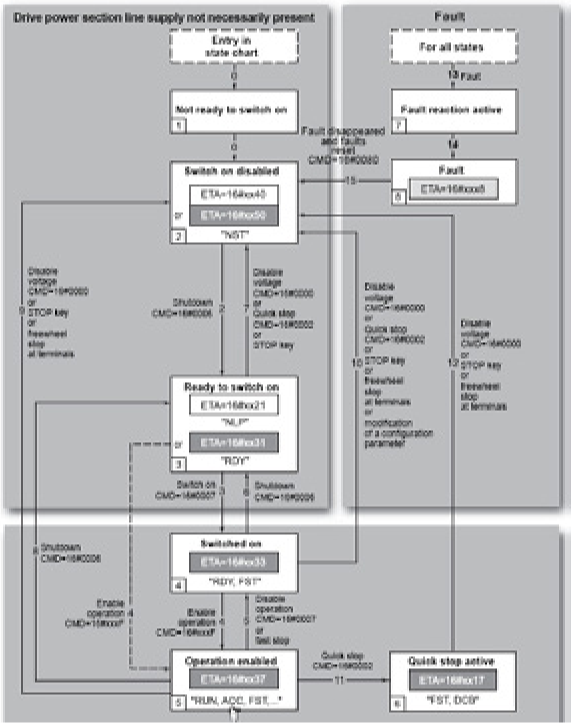
<table>
<tr><td>1</td><td>An example of how to create a function block for the Altivar 71 appears below.

Once the FB has been created, the inputs and outputs are defined.
The internal variables required will also need to be configured.</td><td>FUNCTION_BLOCK ATV71 (* simple function block for ATV71 *)

VAR_INPUT
ATV_IN : atv_data; (* communication from ATV *)
Start : BOOL; (* start drive *)
EStop : BOOL; (* emergency stop, 1=OK, 0=E-Stop *)
Dir : BOOL; (* direction, 0=right, 1=left *)
Ackn : BOOL; (* acknowledge error *)
Speed_Set : INT ; (* speed setpoint *)
CommOK : BOOL; (* communication OK *)
END_VAR

VAR_OUTPUT
ATV_OUT: atv_data; (* communication to ATV *)
Run: BOOL; (* drive is running *)
Mot_ES: BOOL; (* drive emergency stop *)
Err: BOOL; (* drive failure *)
ErrCode: INT; (* drive error code*)
Speed_Act: INT; (* actual drive speed *)
AC_PWR_OK: BOOL; (* AC power present*)
END_VAR</td></tr>
<tr><td>2</td><td>This is what the information will actually look like to the user.</td><td></td></tr>
<tr><td>2</td><td>In the Altivar 71, the relationships between the state transitions are mapped in the state machine.

The operating states are controlled by the user with the control word and monitored with the status word.</td><td></td></tr>
<tr><td>3</td><td>An operating state chart appears opposite.

The ATV71 documentation (Communication parameters) contains a detailed description of the individual operating states.</td><td></td></tr>
</table>

4

State	bit 6 Switch on disabled	bit 5 Quick stop	bit 4 Voltage enabled	bit 3 Fault	bit 2 Operation enabled	bit 1 Switched on	bit 0 Ready to switch on	ETA (W3201) masked by 16#006F (1)
1 - Not ready to switch on	0	x	x	0	0	0	0	-
2 - Switch on disabled	1	x	x	0	0	0	0	16#0040
3 - Ready to switch on	0	1	x	0	0	0	1	16#0021
4 - Switched on	0	1	1	0	0	1	1	16#0023
5 - Operation enabled	0	1	1	0	1	1	1	16#0027
6 - Quick stop active	0	0	1	0	1	1	1	16#0007
7 - Fault reaction active	0	x	x	1	1	1	1	-
8 - Fault	0	x	x	1	0	0	0	16#0008 (2) or 16#0028

The operating states are indicated in bits 0 to 6 of the status word.

The status is read via CANopen and written to the block input.

The status is read first in the FB.

```
(* Set new state of ATV *)
IF ((State_Ctrl AND 16#0F) = 16#08) THEN
    ATV_Error := 1;                    (*FAULT*)
ELSE
 CASE BYTE_TO_INT(State_Ctrl) OF
    16#00: ATV_NoVoltage  := 1;    (*Not ready to switch on*)
    16#40: ATV_Locked     := 1;    (*Switch on disable    - nSt -*)
    16#50: ATV_Locked     := 1;    (*Switch on disable    - nSt -*)
    16#21: ATV_Wait       := 1;    (*Ready to switch on   - nSt -*)
    16#31: ATV_Wait       := 1;    (*Ready to switch on   - nSt -*)
    16#23: ATV_Ready      := 1;    (*Switched on          - nSt -*)
    16#33: ATV_Ready      := 1;    (*Switched on          - nSt -*)
    16#27: ATV_Run        := 1;    (*Operation enabled    - rUn -*)
    16#37: ATV_Run        := 1;    (*Operation enabled    - rUn -*)
    16#07: ATV_Estop      := 1;    (*Quick stop active    - rdY, dCb -*)
    16#17: ATV_Estop      := 1;    (*Quick stop active    - rdY, dCb -*)
 END_CASE;
END_IF;

IF State_Ctrl = 16#40 OR State_Ctrl = 16#21 THEN
    ATV_noAC        := 1;
END_IF;
```

5

Command	Transition address	Final state	bit 7 Fault reset	bit 3 Enable operation	bit 2 Quick stop	bit 1 Enable voltage	bit 0 Switch on	Example value
Shutdown	2, 6, 8	3 - Ready to switch on	x	x	1	1	0	16#0006
Switch on	3	4 - Switched on	x	x	1	1	1	16#0007
Enable operation	4	5 - Operation enabled	x	1	1	1	1	16#000F
Disable operation	5	4 - Switched on	x	0	1	1	1	16#0007
Disable voltage	7, 9, 10, 12	2 - Switch on disabled	x	x	x	0	x	16#0000
Quick stop	11	6 - Quick stop active	x	x	0	1	x	16#0002
	7, 10	2 - Switch on disabled						
Fault reset	15	2 - Switch on disabled	0 → 1	x	x	x	x	16#0080

The VSD is controlled by means of bits 0 to 3 and bit 7 in the control word.

The commands are available at the block output and are transferred to the VSD via CANopen.

The corresponding commands are generated in the FB on the basis of the control settings.

```
(* ATV Set to Wait-Mode      ###     step 2   ### *)
IF ATV_Locked THEN
    ATV_OUT.data := 16#0006;
END_IF;

(* ATV Set to Ready-Mode     ###     step 3   ### *)
IF ATV_Wait THEN
    ATV_OUT.data := 16#0007;
END_IF;

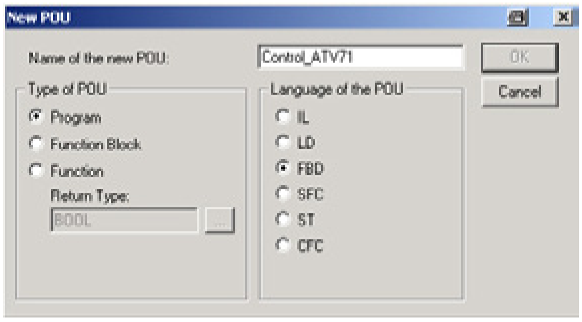
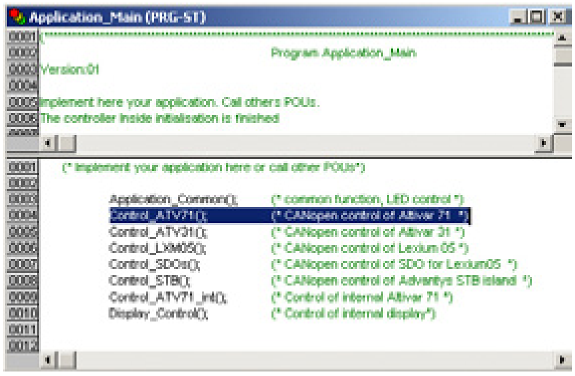
(* ATV Start Operation       ###     step 4   ### *)
IF (ATV_Ready AND Start AND NOT Dir) THEN
    ATV_OUT.data := 16#000F;
ELSIF (ATV_Ready AND Start AND Dir) THEN
    ATV_OUT.data := 16#080F;
END_IF;

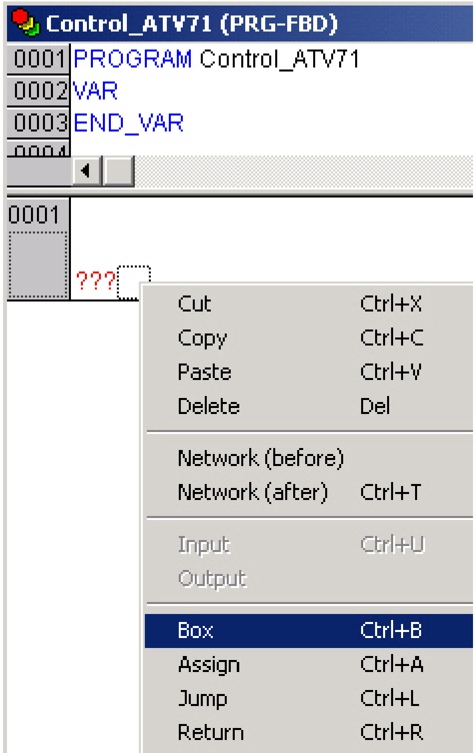
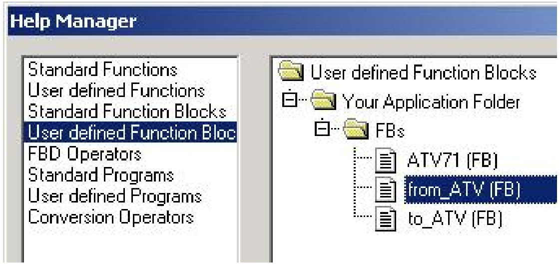
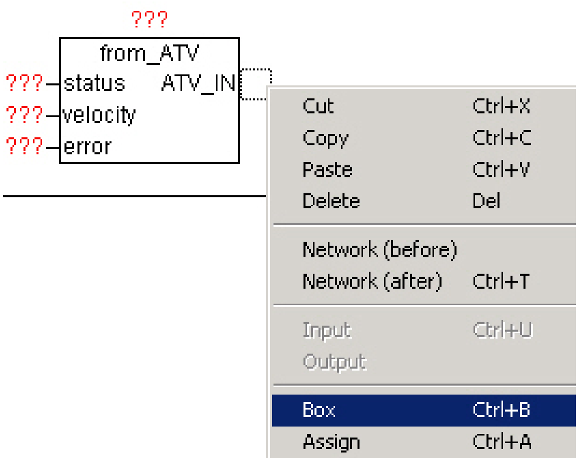
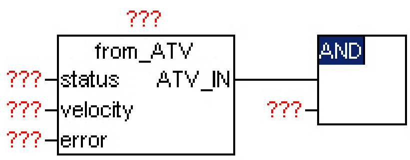
(* ATV in Operation          ###     stay running      ### *)
IF (ATV_Run AND Start AND NOT Dir) THEN
    ATV_OUT.data := 16#000F;
ELSIF (ATV_Run AND Start AND Dir) THEN
    ATV_OUT.data := 16#080F;

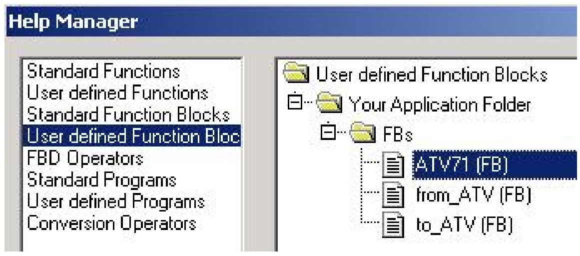
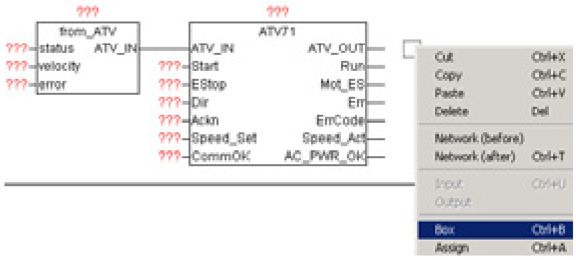
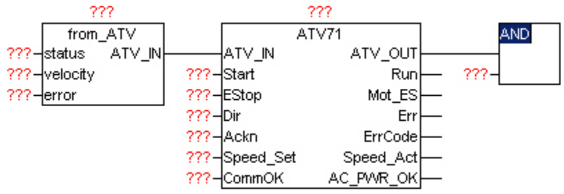
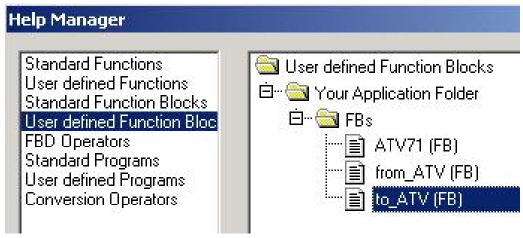
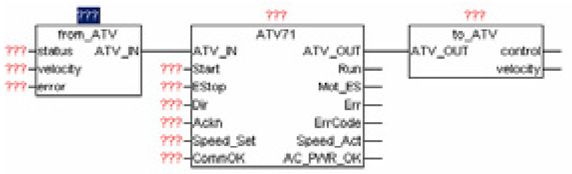
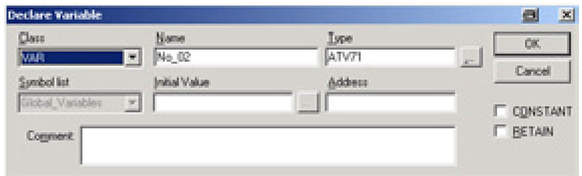
(* ATV Set to Stop           ###     step 5   ### *)
ELSIF (ATV_Run AND NOT Start) THEN
    ATV_OUT.data := 16#0007;
END_IF;
```

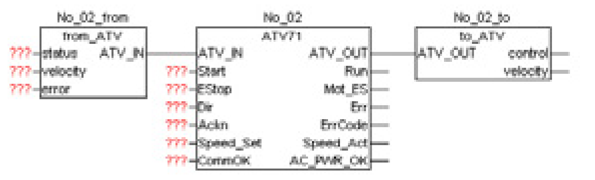
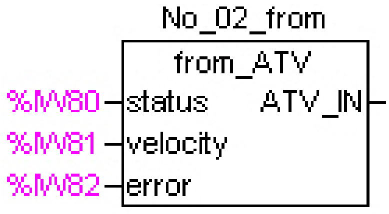
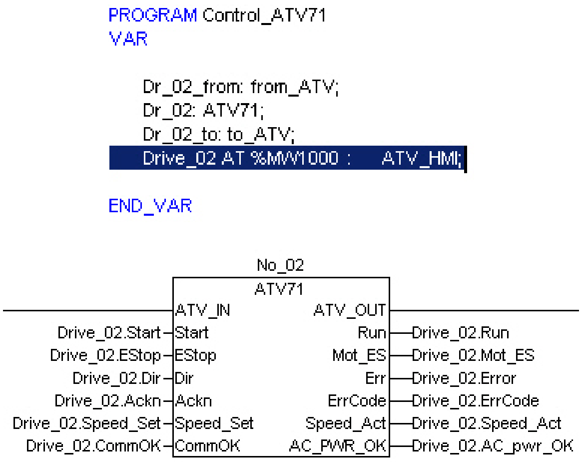
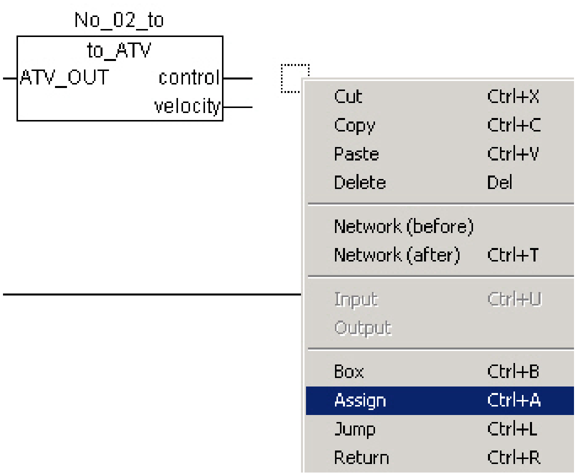
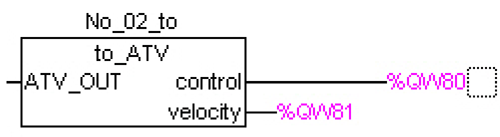
6	Based on the input, the setpoint speed (i.e. the frequency) for the ATV, along with the status information is set and moved to the block outputs.	(* ATV Frequency *) IF Start THEN ATV_OUT.speed := INT_TO_WORD(Speed_Set); ELSE ATV_OUT.speed := 0; END_IF;

Creating a Program Block (FBD) for ATV71

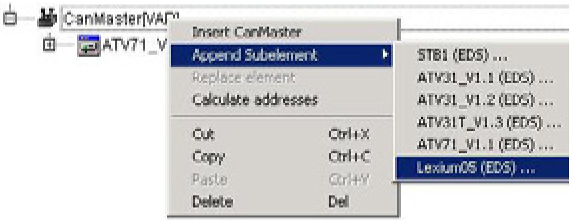
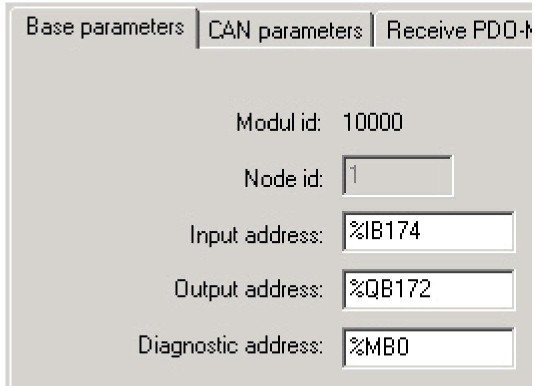
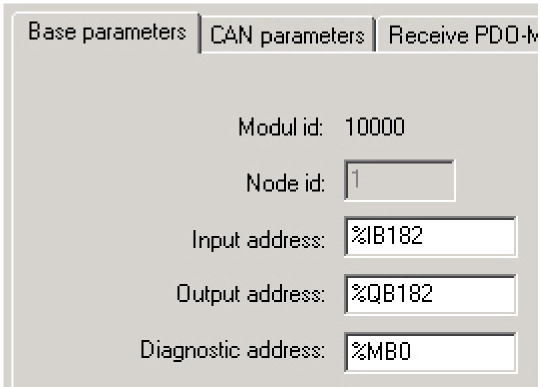
1	In **Your Application Folder**, click with the right mouse button and select: **Add Object...** to create a program block.	
2	Select **Program** under **Type of POU** in the next window. You also need to define the POU name and language. In this case, these are **Control_ATV71** and **FBD** respectively.	
3	In order for the function block to be executed, it must be called by a program block. In this case, it is called by the main program: **Application_Main**. The entry can be made directly or via **F2** (input assistant).	

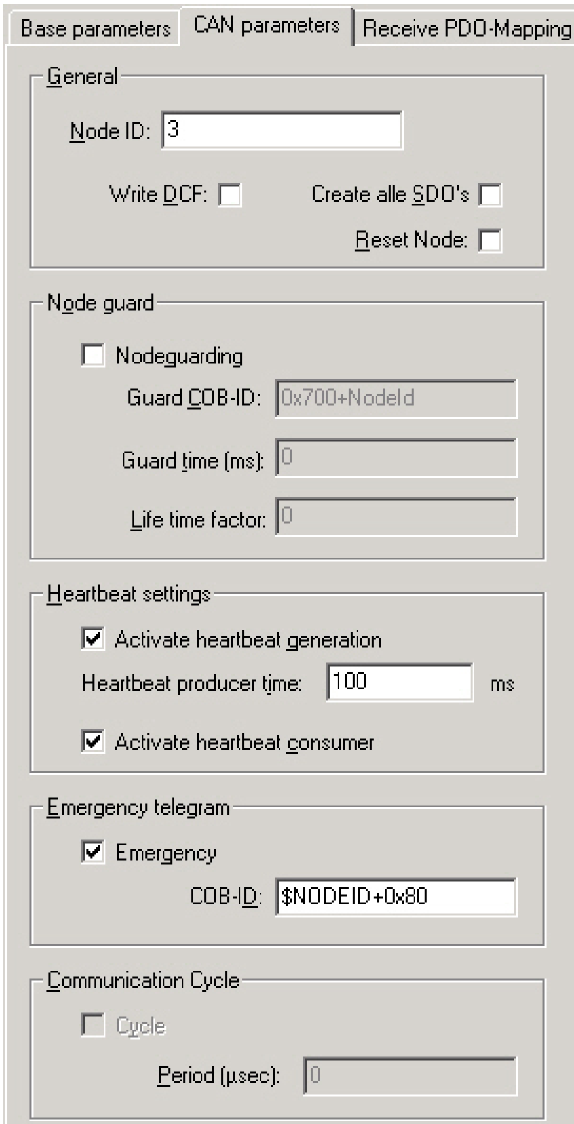
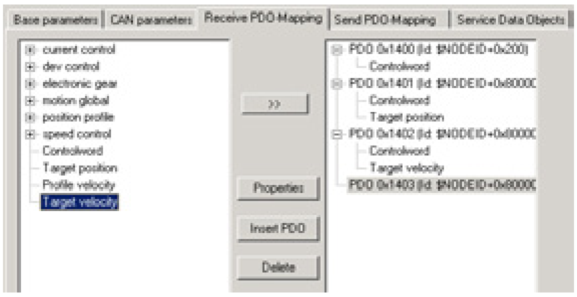
4	Once the program block has been added, you need to add a block placeholder. Do this by right-clicking with the right mouse button and selecting **Block**.	
5	Then use **F2** to select the required block. In this case: the **from_ATV** function block created previously.	
6	Now use the right mouse button to connect another **block** at the FB's output pin.	
7	Use **F2** to specify.	

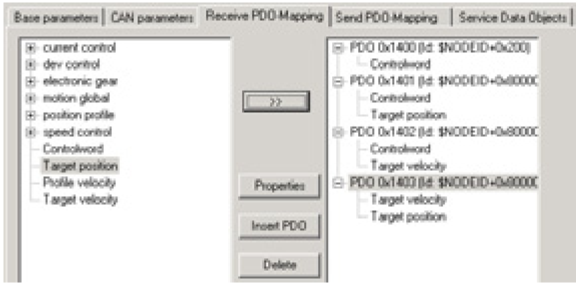
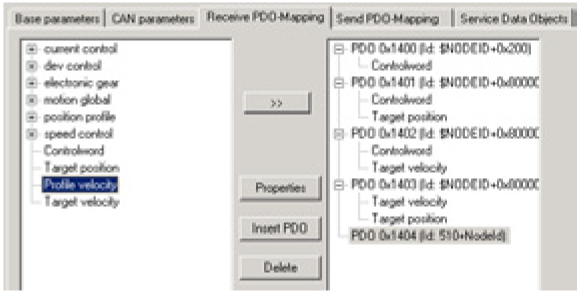
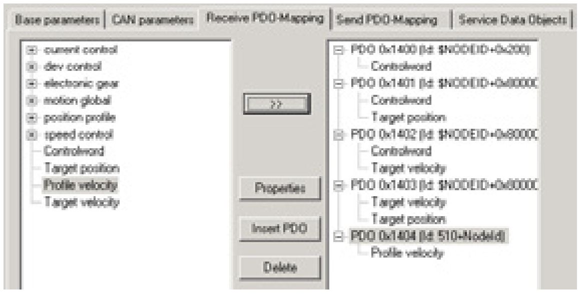
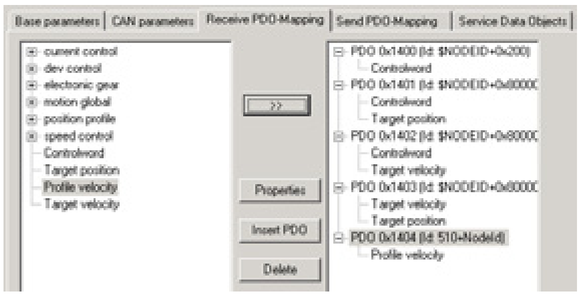
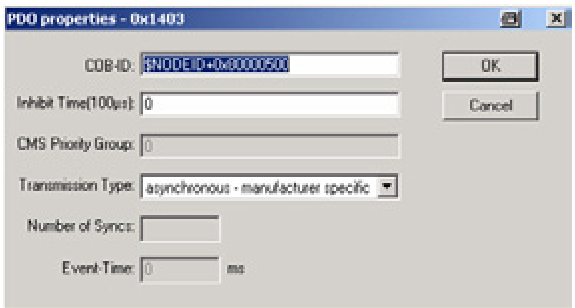
8	And select FB **ATV71**.	
9	Add another **block.**	
10	And use **F2**	
11	to select FB **to_ATV**.	
12	Each FB instance has an associated designator (instance name) As with variables, instances are declared locally or globally by specifying the FB name as the designator type.	
13	Select the designator (**???**) for the FB and enter the name **No_02_from.** Once you have confirmed the name, this window will appear. Click **OK** to exit**.** The entry will be made in the declaration section automatically.	
14	Now do the same for the FB **ATV71** and	
15	and **FB to_ATV** instances.	

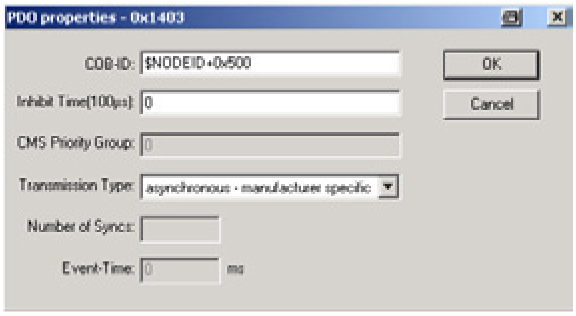
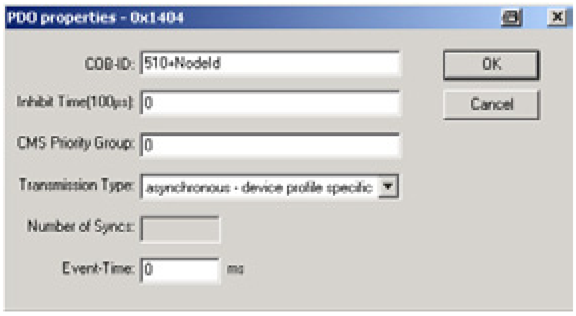
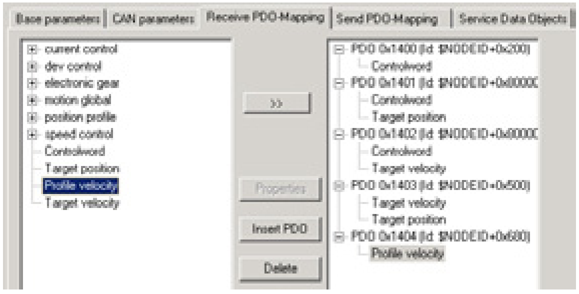
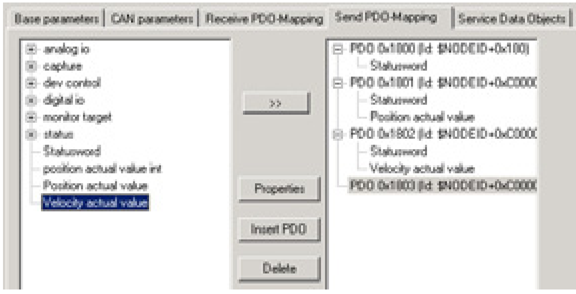
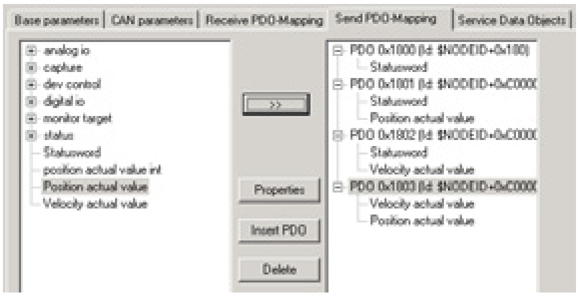
16	Instanced FBs.	
17	Now enter the input addresses. For information about addresses, see also Communication. You can also connect variables here.	
18	You also need to connect the inputs and outputs to FB **ATV71**. In this example, the variables are being connected by the HMI. Thus, the drive is controlled directly by the HMI. The variable is a structure and is assigned an address in the declaration section. This is what the interface with the actual application will eventually look like.	
19	To connect the output address to the final block, right-click with the mouse and select **Assign.**	
20	And then enter the output addresses. For information about addresses, see also Communication.	

Linking Lexium05 for CANopen

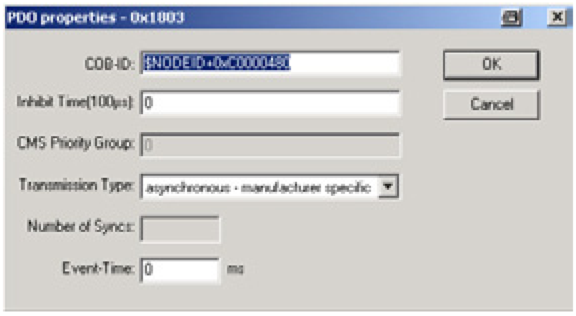
1	Right-click with the mouse on **CanMaster** and select **Lexium05** from the **Append Subelement** list. Alternatively, this function can be accessed via the menu by selecting **Insert -> Append Subelement.**	
2	Once you have selected the Lexium05, a properties dialog box will appear on the right-hand side. On opening the **Base parameters** tab you are offered some default addresses. These must be changed. Enter **%IB182** as the input address and **%QB182** as the output address for the first Lexium05. (See the list in the Communication chapter for more information.)	

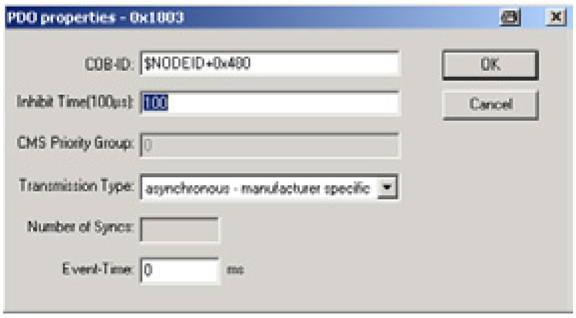
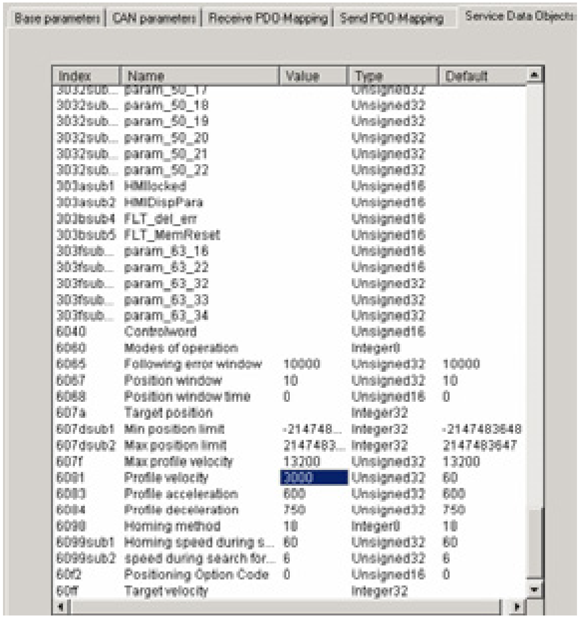
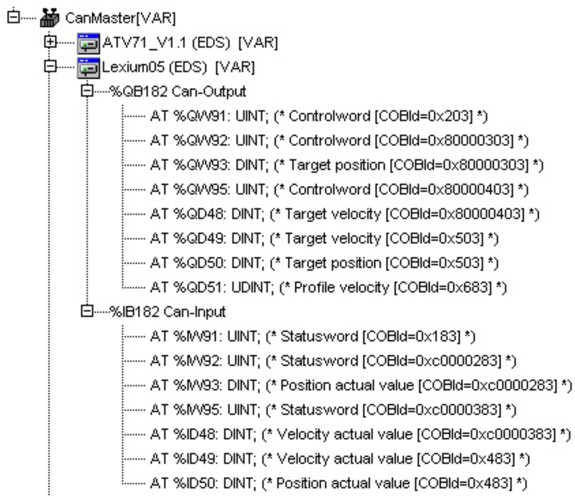
<table>
<tr>
<td>3</td>
<td>Enter the CANopen address 3 in the Node ID field on the CAN parameters tab for the first Lexium05. Addresses 4 to 7 are used for the other servos.
Nodeguarding is unchecked; Heartbeat and Emergency are checked.
The Heartbeat producer time is 100 ms.</td>
<td>
</td>
</tr>
<tr>
<td>4</td>
<td>On the Receive PDO-Mapping tab (Drive <- PLC), the default setting for the first three PDOs is accepted.
The setting for the fourth PDO is expanded.</td>
<td>
</td>
</tr>
</table>

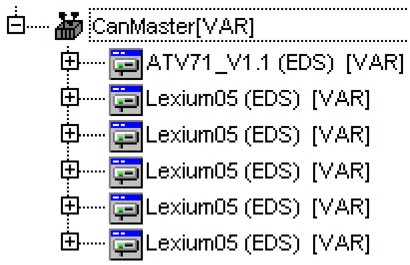
5	Select the **target velocity** and **target position** setpoints on the left-hand side (one after the other) and use **>>** to transfer them to the right-hand column. This uses up the maximum number of 8 bytes (2 doublewords). Note: The second and third PDOs are not used, as otherwise the control word would be transferred more than once. This could result in mixups.	
6	A fifth PDO is required for the **Profile velocity**. To add this PDO, click on **Insert PDO**.	
7	Now use **>>** to insert the Profile velocity under this PDO. **Note:** The profile velocity is not required in this application. It is merely intended to illustrate a new PDO.	
8	The fourth PDO is deactivated by default. You can activate it by selecting it in the right-hand field and clicking on **Properties** to open the Properties window. Change **0x80000500** to **0x00000500** or **0x500**. If the most significant bit = 1, the PDO is deactivated. A 0 means it is activated.	

9	You will need to change the proposed COB-ID for the fifth PDO. According to the CANopen specification, the range 680 ... 6FF is reserved for PDOs > 4. Change **510** to **0x680**, plus NodeID.	 
10	The following should now be visible in the window.	
11	The process for dealing with the **Send PDO-Mapping** tab (Drive -> PLC) is extremely similar. Leave the first three PDOs exactly as they are.	
12	Expand the fourth PDO by adding **Velocity actual value** and **Position actual value**.	

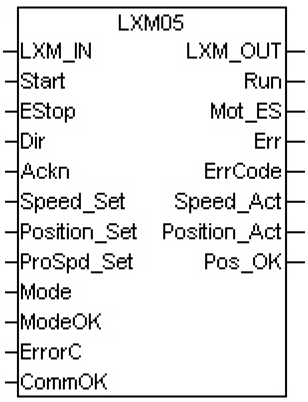
<table>
<tr>
<td>13</td>
<td>The fourth PDO is deactivated by default.
Change
0x80000500 to
0x480.

You also need to enter an Inhibit Time of 10 ms to prevent the bus from becoming overloaded.
You will need to enter the value 100, because the multiplication factor is 100µs (100 x 100 µs = 10ms).</td>
<td>


</td>
</tr>
<tr>
<td>14</td>
<td>You can also change the default start parameters using the Service Data Objects tab. Do this by entering the new value in the Value column. In this example, the Profile velocity value has been changed from 60 to 3000.</td>
<td>
</td>
</tr>
<tr>
<td>15</td>
<td>The following should now be visible in the PLC Configuration window.
This screenshot lists the start address and COB ID for the individual parameters.</td>
<td>
</td>
</tr>
</table>

16	Connect the other four Lexium05 servo drives in the same way.	
17	When the PLC starts up, the **ControllerInside_Start** program is called once and the connected CANopen nodes are declared. You should add the nodes with the addresses **3 to 7** here.	(*put here the NodeID of the slaves you configured in Plc configuration*) CanOpenAddSlave(CanOpenNodeID:= 2); CanOpenAddSlave(CanOpenNodeID:= 3); CanOpenAddSlave(CanOpenNodeID:= 4); CanOpenAddSlave(CanOpenNodeID:= 5); CanOpenAddSlave(CanOpenNodeID:= 6); CanOpenAddSlave(CanOpenNodeID:= 7);

Special features in function block (ST) for LXM05

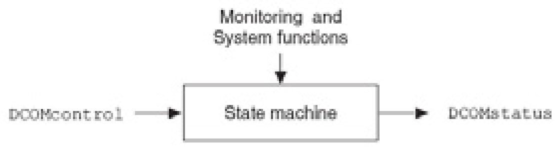
1	Compared with the VSDs, the Lexium05 offers a greater range of operating modes. This FB is used to implement the velocity and positioning modes. For precise details of operating modes, please refer to the Lexium05 documentation. Once the FB has been created, the first thing you need to do is to define the **inputs and outputs**. Other essential internal variables will also need to be configured.	FUNCTION_BLOCK LXM05 (* function block for Lexium05 *) VAR_INPUT LXM_IN: LXM_data; (* communication from Lexium *) Start: BOOL; (* start servo *) EStop : BOOL; (* emergency stop, 1=OK, 0=E-Stop *) Dir : BOOL; (* direction, 0=right, 1=left *) Ackn : BOOL; (* acknowledge error *) Speed_Set : DINT; (* speed setpoint *) Position_Set: DINT; (* position setpoint*) ProSpd_Set: UDINT; (* profil velocity/speed setpoint*) Mode: BYTE; (* mode for the servo, 1=position mode, 3=velocity/speed mode *) ModeOK: BOOL; (* mode is transferd to the servo *) ErrorC: WORD; (* error code for servo via SDO *) CommOK : BOOL; (* communication OK *) END_VAR VAR_OUTPUT LXM_OUT: LXM_data; (* communication to Lexium *) Run: BOOL; (* drive is running *) Mot_ES: BOOL; (* drive emergency stop *) Err: BOOL; (* drive failure *) ErrCode: INT; (* drive error code*) Speed_Act: DINT; (* actual servo speed *) Position_Act: DINT; (* actual servo position *) Pos_OK: BOOL; (* position is reached *) END_VAR
2	This is what the LXM05 FB will actually look like to the user.	

3

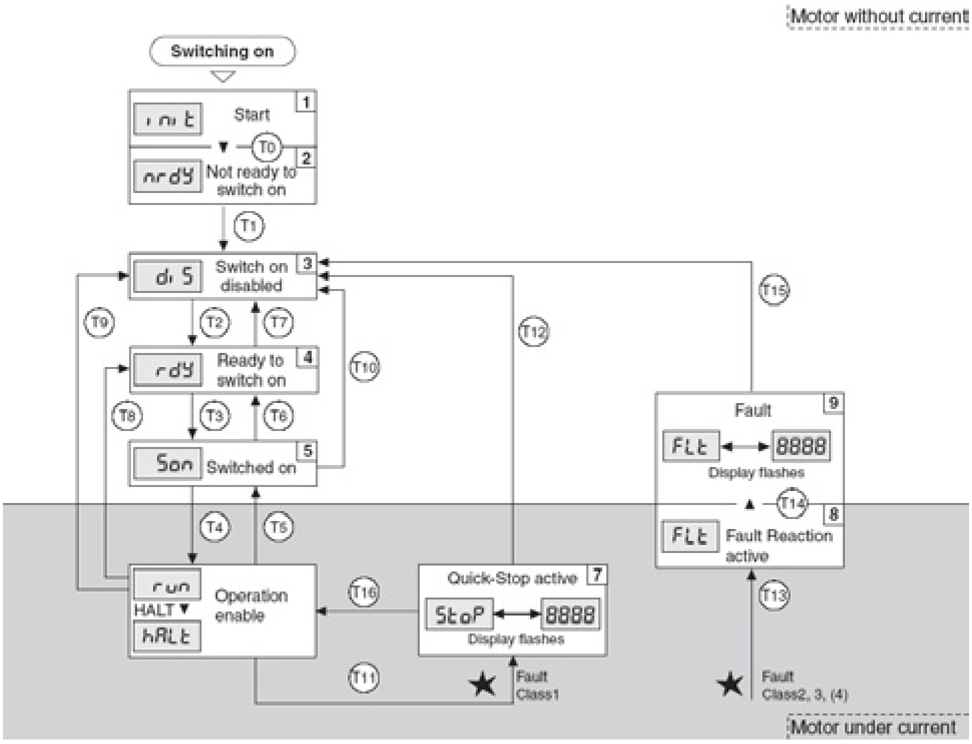
The relationships between the state transitions are mapped in the state machine.

The operating states are influenced by the user with the control word (**DCOMcontrol**) and monitored with the status word (**DCOMstatus**).

Below is a chart showing the operating states.



4



5

The operating state is represented by bits 0, 1, 2, 3, 5 and 6.



Status	Bit 6,Switch Ondisable	Bit 5, Quick-Stop	Bit 3,Fault	Bit 2, OperationE-NABLE	Bit 1, Switch On	Bit 0, Ready toSwitch On
2: Not ready to switch on	0	X	0	0	0	0
3: Switch on disabled	1	X	0	0	0	0
4: Ready to switch on	0	1	0	0	0	1
5: Switched on	0	1	0	0	1	1
6: Operation enable	0	1	0	1	1	1
7: Quick Stop active	0	0	0	1	1	1
9: Fault	0	X	1	1	1	1

6

The operating state is read via CANopen and written to the block input.
The status is detected at the start of the FB.

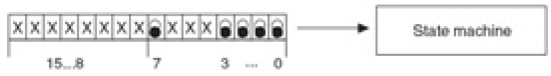
```
(* State Machine  -  status  *)

cState6 := LXM_IN.data AND 2#01101111;
cState5 := LXM_IN.data AND 2#01001111;

IF       cState5 = 0 THEN
            State := 2;                          (*  Not ready to switch on    nrdY   *)
ELSIF    cState5 = 16#40 THEN
            State := 3;                          (*  Switch on disabled        diS   *)
ELSIF    cState6 = 16#21 THEN
            State := 4;                          (*  Ready to switch on        rdY   *)
ELSIF    cState6 = 16#23 THEN
            State := 5;                          (*  Switched on               Son   *)
ELSIF    cState6 = 16#27 THEN
            State := 6;                          (*  Operation enable          run   *)
ELSIF    cState6 = 16#07 THEN
            State := 7;                          (*  Qiuck Stop active         StoP   *)
ELSIF    cState5 = 16#0F THEN
            State := 9;                          (*  Fault                     FLt   *)
ELSIF    cState5.3 THEN
            State := 9;                          (*  Fault                     FLt   *)
END_IF
```

7

The servo drive is controlled via bits 0, 1, 2, 3, and 7.



Fieldbus command	state transitions	Change of state to	Bit 7, Reset Fault	Bit 3, Enable operation	Bit 2, Quick-Stop	Bit 1, Disable Voltage	Bit 0, Switch On
Shutdown	T2, T6, T8	4: Ready to switch on	X	X	1	1	0
Switch On	T3	5: Switched on	X	X	1	1	1
Disable Voltage	T7, T9, T10, T12	3: Switch on disabled	X	X	X	0	X
Quick Stop	T7, T10T11	3: Switch on disabled7: Quick Stop active	X	X	0	1	X
Disable Operation	T5	5: Switched on	X	0	1	1	1
Enable operation	T4, T16	6: Operation enable	X	1	1	1	1
Fault Reset	T15	3: Switch on disabled	0 -> 1	X	X	X	X

8

The corresponding control settings are generated on the basis of the commands at the block input. The block output then transmits these to the servo drive via CANopen. The same applies to the setpoints. Status information is also made available at the block output.

```
(* State machine  -  control  *)

IF CommOK AND NOT ModeOK AND NOT Start THEN
    PowerUP := TRUE;
ELSE
    PowerUP := FALSE;
END_IF

IF State=2 THEN
    LXM_OUT.data := 2#00000000;

ELSIF (State=6 OR State=5 OR State=4) AND NOT EStop THEN
    LXM_OUT.data := 2#00000010;                    (* T7, T10, T11 - Quick Stop *)

ELSIF State=3 AND ((Start AND ModeOK) OR PowerUP) THEN
    LXM_OUT.data := 2#00000110;                    (* T2 - Shutdown *)

ELSIF State=4 AND ((Start AND ModeOK) OR PowerUP) THEN
    LXM_OUT.data := 2#00000111;                    (* T3 - Switch on *)

ELSIF State=5 AND ((Start AND ModeOK) OR PowerUP) THEN
    LXM_OUT.data := 2#00001111;                    (* T4 - Enable operation *)

ELSIF State=6 AND ((Start AND ModeOK) OR PowerUP) THEN
    LXM_OUT.data := 2#00001111;                    (* T4 - Enable operation *)

ELSIF State=7 AND ((Start AND ModeOK) OR PowerUP) AND EStop AND Ackn THEN
    LXM_OUT.data := 2#00001111;                    (* T16 - Enable operation *)

ELSIF (State=4 OR State=5 OR State=6 OR State=7) AND (NOT Start AND NOT PowerUP) THEN
    LXM_OUT.data := 2#00000000;                    (* T7, T9, T10, T12 - Disable Voltage *)

ELSIF State=9 AND (Ackn OR PowerUP) THEN
    LXM_OUT.data := 2#10000000;                    (* T15 - Fault reset *)

END_IF
```

<table>
<tr>
<td>9</td>
<td>As certain data cannot be made available via the PDOs, it is read in/written via SDOs.

Execution is handled by the Control_SDOs program block. The individual servo drives and SDOs are processed one after the other, as only one SDO may be active at any one time.</td>
<td><pre>
IF node_count < 6 AND NOT SDO_EN THEN
    IF DriveLXM[node_count].CommOK THEN
        Node_id   := DriveLXM[node_count].Node_id;
        Mode   := DriveLXM[node_count].Mode;
        SDO_EN := TRUE;
    ELSE
        node_count := node_count + 1;
    END_IF
END_IF

IF SDO_EN AND SDO_OK THEN
    node_count := node_count + 1;
    SDO_EN := FALSE;
    SDO_OK := FALSE;
END_IF
</pre></td>
</tr>
<tr>
<td>10</td>
<td>First of all, the operating mode entered in the servo drive is read out.

If the operating mode does not match the mode specified by the operator/program, this information is forwarded to the FB by means of the xxx.ModeOK=0 structure element.

The operating state can only be changed in State 6 = run. The FB uses the structure element referred to above to start the servo drive, but without setpoints.</td>
<td><pre>
                    (*   Read Mode from Drive   *)
IF SDO_count = 0 THEN
    IF NOT Read1.CONFIRM THEN
        R1_start := TRUE;
    ELSE
        R1_start := FALSE;
        SDO_count := 1;
        IF Read1_data[1] = Mode THEN
            DriveLXM[node_count].ModeOK := TRUE;
        ELSE
            DriveLXM[node_count].ModeOK := FALSE;
        END_IF
    END_IF

    Read1(DEVICE:= Node_id,
            INDEX:= 16#6061,
            SUBINDEX:= 0,
            ENABLE:= R1_start,
            DATA=> Read1_data);
END_IF
</pre></td>
</tr>
<tr>
<td>11</td>
<td>As soon as the state is achieved, the current operating mode is transmitted.</td>
<td><pre>
IF SDO_count = 2 THEN
    IF Read1_data[1] <> Mode AND NOT Write1.CONFIRM AND DriveLXM[node_count].Run THEN
        W1_start := TRUE;
        Write1_data[1] := Mode;
    ELSE
        W1_start := FALSE;
        SDO_count := 0;
        SDO_OK := TRUE;
    END_IF

    Write1(DEVICE:= Node_id,
        INDEX:= 16#6060,
        SUBINDEX:= 0,
        ENABLE:= W1_start,
        DATA:= Write1_data,
        DATALENGTH:= 1);
END_IF
</pre></td>
</tr>
<tr>
<td>12</td>
<td>The error word is also read out and forwarded to the FB.</td>
<td><pre>
                 (*   Read Error from Drive   *)
IF SDO_count = 1 THEN
    IF NOT Read2.CONFIRM THEN
        R2_start := TRUE;
    ELSE
        R2_start := FALSE;
        SDO_count := 2;
        Help := Read2_data[2];
        DriveLXM[node_count].ErrorC := SHL(Help,8) OR Read2_data[1];
    END_IF

    Read2(DEVICE:= Node_id,
            INDEX:= 16#603F,
            SUBINDEX:= 0,
            ENABLE:= R2_start,
            DATA=> Read2_data);
END_IF
</pre></td>
</tr>
</table>

Creating a Program Block (FBD) for LXM05

1

The Lexium05 servo drives are controlled by means of the **Control_LXM05** program block.
The procedure for creating and parameterizing the program block is exactly the same as for the Altivar 71 and 31 drives, so please refer to the relevant description and proceed accordingly.

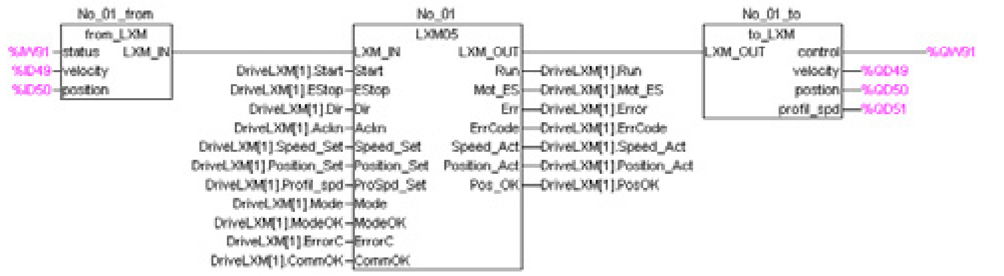
2

First, the CANopen status of each node is copied to the relevant structure. Then a check is performed to ensure that the **status** is **Operational** (=OK; Value=5).

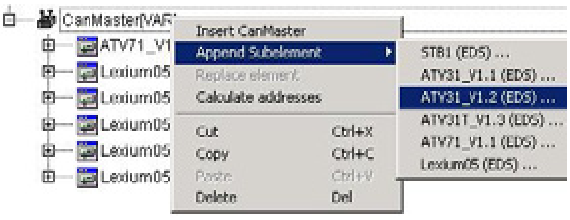
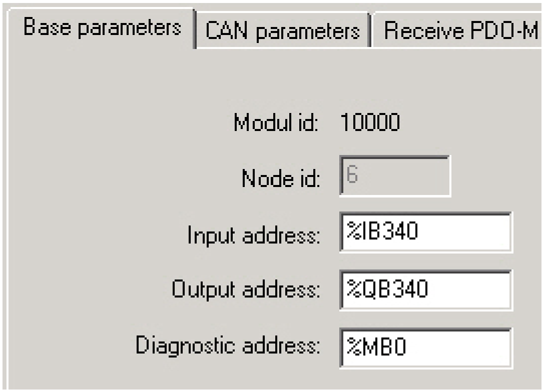
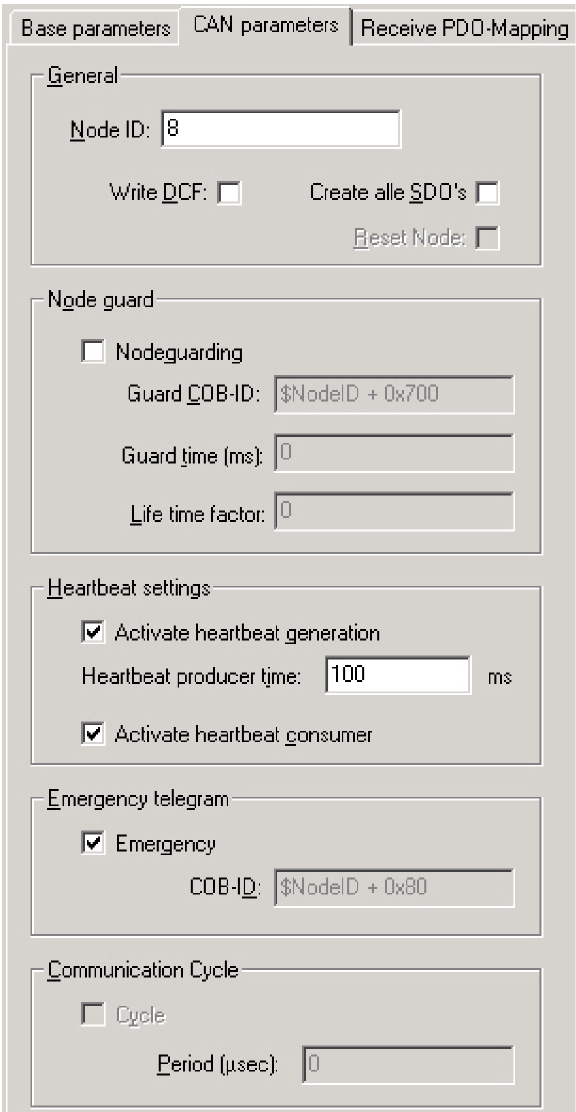


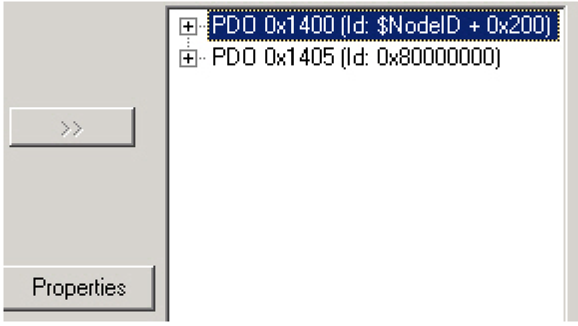
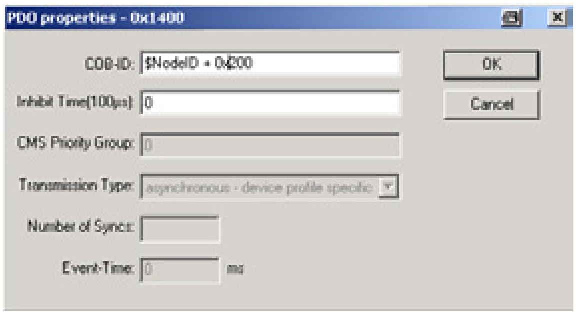
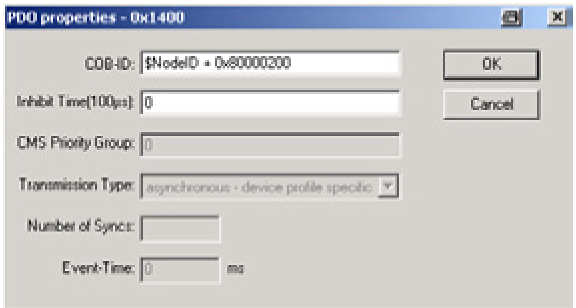
3

Then, the integrated FBs for a Lexium05 servo drive are displayed.
For information about input and output addresses, see Communication.
In this example, the HMI variables are connected directly to the FB so that the servo drive can be controlled by the HMI. This is what the interface with the actual application will eventually look like.



Linking Altivar 31 for CANopen

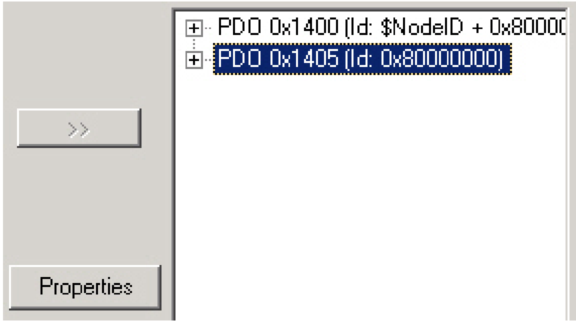
1	Right-click with the mouse on **CanMaster** and select **ATV31_V1.2** from the Append Subelement list. Alternatively, this function can be accessed via the menu by selecting **Insert -> Append Subelement.**	
2	Once you have selected the Altivar 31, a properties dialog box will appear on the right-hand side. On the **Base parameters** tab, enter **%IB340** as the input address and **%QB340** as the output address for the first ATV31. (See the list in the Communication chapter for more information.)	
3	Enter the **CANopen address 8** in the Node ID field on the **CAN parameters** tab for the first ATC31. Addresses 9 to 13 are used for the other drives. Nodeguarding is unchecked; **Heartbeat** is set to **100 ms** and Emergency is checked.	

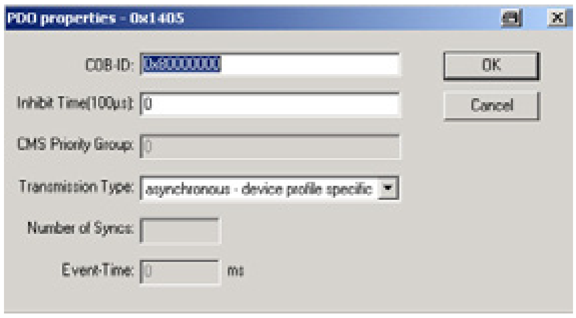
4	On the **Receive PDO-Mapping** tab (Drive <- PLC), the 1st and 6th PDOs are displayed.	
5	The first PDO is activated by default. You can deactivate it by selecting it in the right-hand field and clicking on **Properties** to open the Properties window. Change: **0x200** to **0x80000200**. The most significant bit is used to activate (=0) or deactivate (=1) a PDO.	

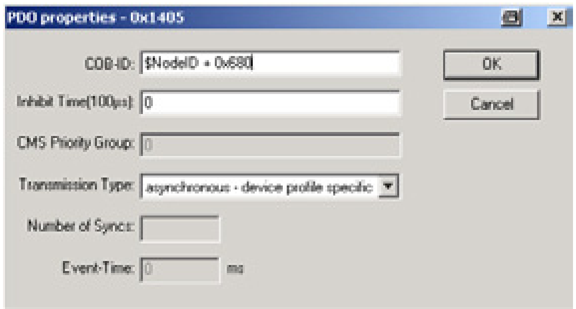
<table>
<tr>
<td>6</td>
<td>The sixth PDO is deactivated by default.

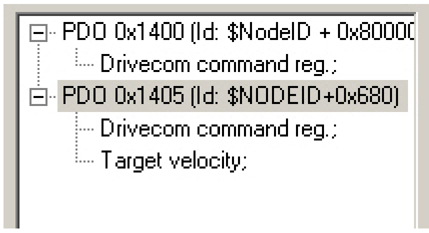
You can activate it by selecting it in the right-hand field and clicking on Properties to open the Properties window.

According to the CANopen specification, the range 680 ... 6FF is reserved for PDOs > 4.

Change
    0x80000000
to
    0x 0x680 + NodeID.</td>
<td>






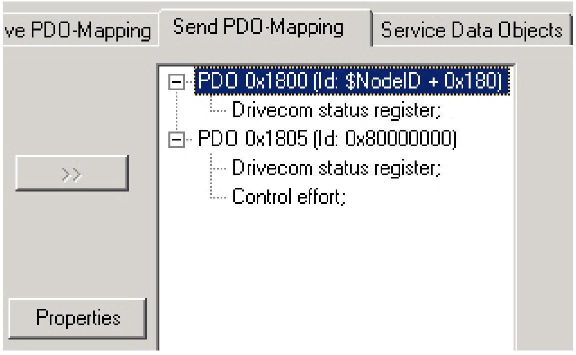
</td>
</tr>
<tr>
<td>7</td>
<td>The following should now be displayed.</td>
<td>


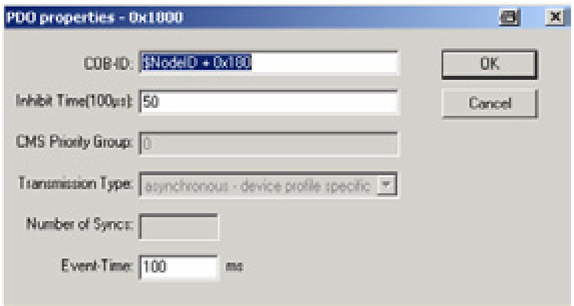
</td>
</tr>
</table>

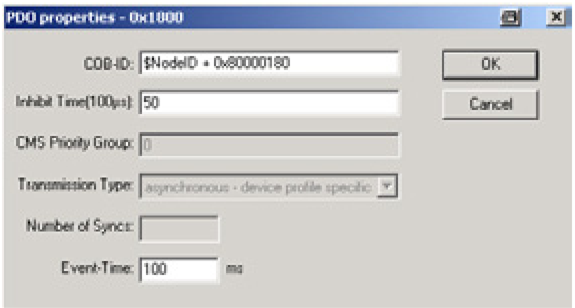
8

The process for dealing with the **Send PDO-Mapping** tab (Drive->PLC) is very similar.

Deactivate the first PDO, changing
0x180
to
0x80000180



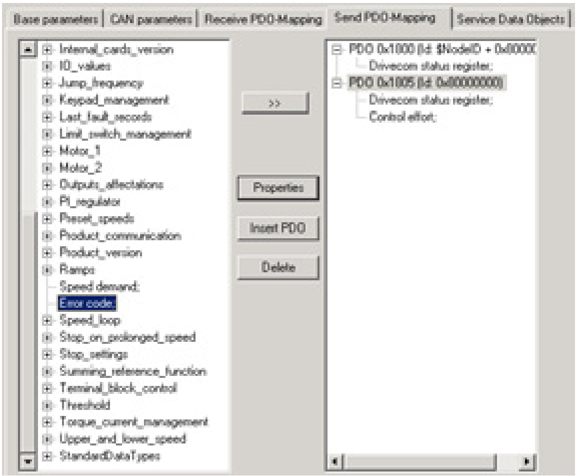




9

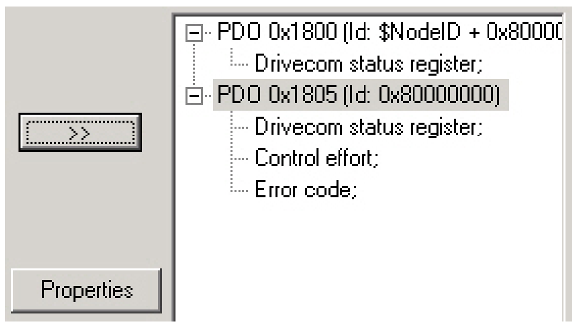
Expand the sixth PDO by adding the **Error code** entry.

Do this by selecting it in the left-hand window and then appending it by means of **>>.**



10

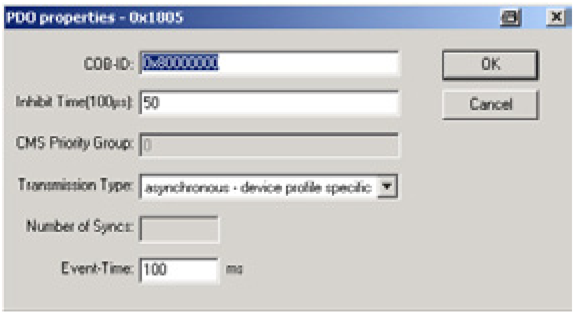
Thus, the following entries should now appear under the sixth PDO.

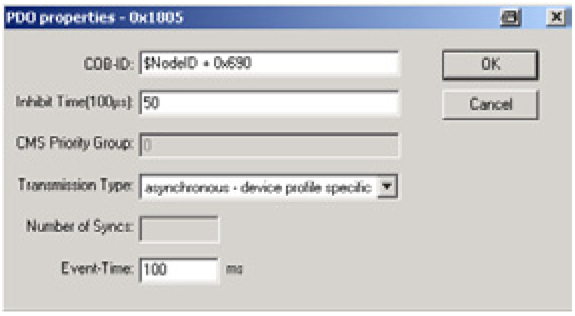


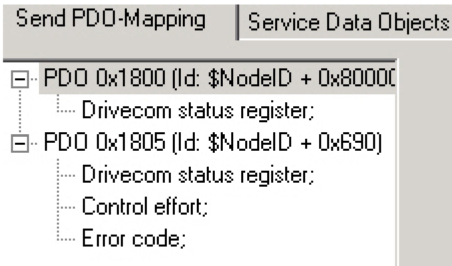
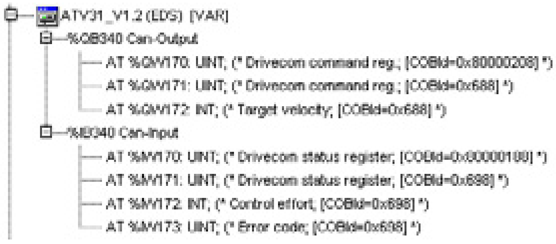
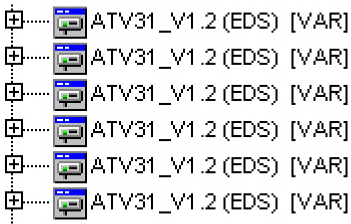
<table>
<tr><td>11</td><td>The sixth PDO is deactivated by default.

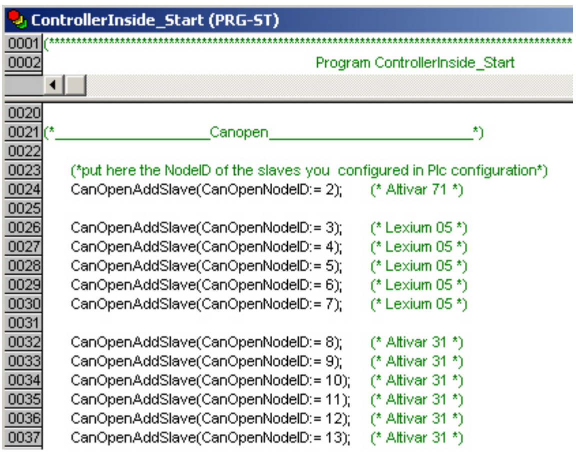
Change
0x80000000
to
0x690 + NodeID.

You also need to enter an **Inhibit Time** to prevent the bus from becoming over-loaded.

Enter a value of **50.**</td><td>




</td></tr>
<tr><td>12</td><td>The following should now be visible in the PLC Configuration window.
This screenshot lists the start address and COB ID for the individual parameters.</td><td>
</td></tr>
<tr><td>13</td><td>Connect the other five Altivar 31 drives in the same way.</td><td>
</td></tr>
<tr><td>14</td><td>When the PLC starts up, the **ControllerInside_Start** program is called once and the connected CANopen nodes are declared.

You should add the nodes with the addresses **8 to 13** here.</td><td>
</td></tr>
</table>

Special features in function block (ST) for ATV31

1

The FB for the Altivar 31 differs only slightly from the Altivar 71 FB.

The status and control settings are adapted. However, externally there is no discernible difference.

First of all, define the FB inputs and outputs along with the other necessary variables.

```
FUNCTION_BLOCK ATV31    (* simple function block for ATV31 *)

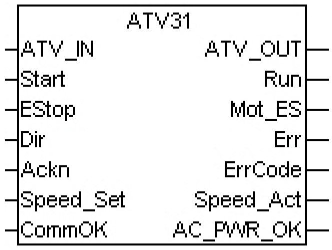
VAR_INPUT
    ATV_IN :        atv_data;   (* communication from ATV *)
    Start :         BOOL;       (* start drive *)
    EStop :         BOOL;       (* emergency stop, 1=OK, 0=E-Stop *)
    Dir :           BOOL;       (* direction, 0=right, 1=left *)
    Ackn :          BOOL;       (* acknowledge error *)
    Speed_Set :     INT ;       (* speed setpoint *)
    CommOK :        BOOL;       (* communication OK *)
END_VAR

VAR_OUTPUT
    ATV_OUT:        atv_data;   (* communication to ATV *)
    Run:            BOOL;       (* drive is running *)
    Mot_ES:         BOOL;       (* drive emergency stop *)
    Err:            BOOL;       (* drive failure *)
    ErrCode:        INT;        (* drive error code*)
    Speed_Act:      INT;        (* actual drive speed *)
    AC_PWR_OK:      BOOL;       (* AC power present*)
END_VAR
```

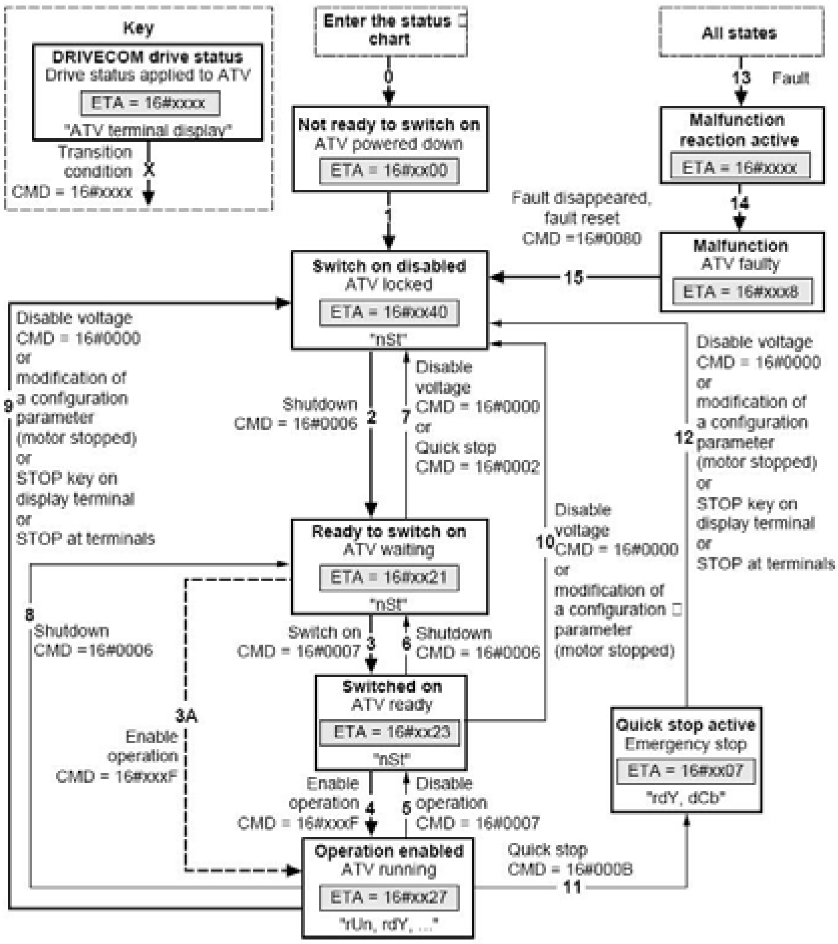
2

This is what the function block will actually look like to the user.

Below is a chart showing the operating states.



3



4

Examples:
ETA = 16#0627 : Normal stop or
□ Forward operation, speed reached
ETA = 16#8627 : Reverse operation, speed reached
ETA = 16#0227 : Forward operation, ACC or DEC
ETA = 16#8227 : Reverse operation, ACC or DEC

Examples (default configuration):
CMD = 16#000F : Forward operation
CMD = 16#080F : Reverse operation
CMD = 16#100F : Stop (configured by "Stt")
CMD = 16#200F : DC injection stop
CMD = 16#400F : Fast stop

5

State	bit 6 Switch on disabled	bit 5 Quick stop	bit 3 Malfunction	bit 2 Operation enabled	bit 1 Switched on	bit 0 Ready to switch on	ETA (W3201) masked by 16#006F
Not ready to switch on	0	x	0	0	0	0	16#0000 16#0020
Switch on disabled	1	x	0	0	0	0	16#0040 16#0060
Ready to switch on	0	1	0	0	0	1	16#0021
Switched on	0	1	0	0	1	1	16#0023
Operation enabled	0	1	0	1	1	1	16#0027
Malfunction	0	x	1	0	0	0	16#0008 16#0028
Malfunction reaction active	0	x	1	1	1	1	16#000F 16#002F
Quick stop active	0	0	0	1	1	1	16#0007

The status is scanned by the VSD via CANopen and connected to the block input.

The status word is monitored in the FB for the purpose of subsequent execution.

```
(* Set new state of ATV *)
IF ((State_Ctrl AND 16#0F) = 16#08) THEN
    ATV_Error := 1;                     (*FAULT*)
ELSE
 CASE BYTE_TO_INT(State_Ctrl) OF
    16#00: ATV_NoVoltage  := 1;        (*Not ready to switch on*)
    16#20: ATV_NoVoltage  := 1;        (*Not ready to switch on*)
    16#40: ATV_Locked     := 1;        (*Switch on disable     - nSt -*)
    16#60: ATV_Locked     := 1;        (*Switch on disable     - nSt -*)
    16#21: ATV_Wait       := 1;        (*Ready to switch on    - nSt -*)
    16#23: ATV_Ready      := 1;        (*Switched on           - nSt -*)
    16#27: ATV_Run        := 1;        (*Operation enabled     - rUn -*)
    16#07: ATV_Estop      := 1;        (*Quick stop active     - rdY, dCb -*)
    16#0F: ATV_Error      := 1;        (*Fault*)
    16#2F: ATV_Error      := 1;        (*Fault*)
 END_CASE;
END_IF;

IF State_Ctrl.4 THEN
    ATV_noAC      := 1;
END_IF;
```

6

Command	Transition address	Final state	bit 7 Reset	bit 3 Enable operation	bit 2 Quick stop	bit 1 Disable voltage	bit 0 Switch on	Typical value of CMD (W8501)
Shut down	2, 6, 8	Ready to switch on	x	x	1	1	0	16#0006
Switch on	3	Switched on	x	x	1	1	1	16#0007
Enable operation	4	Operation enabled	x	1	1	1	1	16#000F
Disable operation	5	Switched on	x	0	1	1	1	16#0007
Disable voltage	7, 9, 10, 12	Switch on disabled	x	x	x	0	x	16#0000
Quick stop	11	Quick stop active	x	x	0	1	x	16#000B or 16#0002
	7, 10	Switch on disabled						
Fault reset	15	Switch on disabled	0 → 1	x	x	x	x	16#0080

The control word is generated on the basis of the commands. The block output then forwards it to the VSD via CANopen.

```
(* ATV Set to Wait-Mode     ###     step 2   ### *)
IF ATV_Locked THEN
     ATV_OUT.data := 16#0006;
END_IF;

(* ATV Set to Ready-Mode    ###     step 3   ### *)
IF ATV_Wait THEN
     ATV_OUT.data := 16#0007;
END_IF;

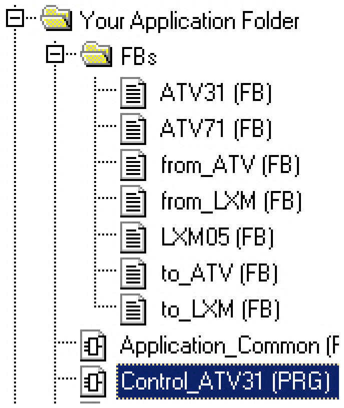
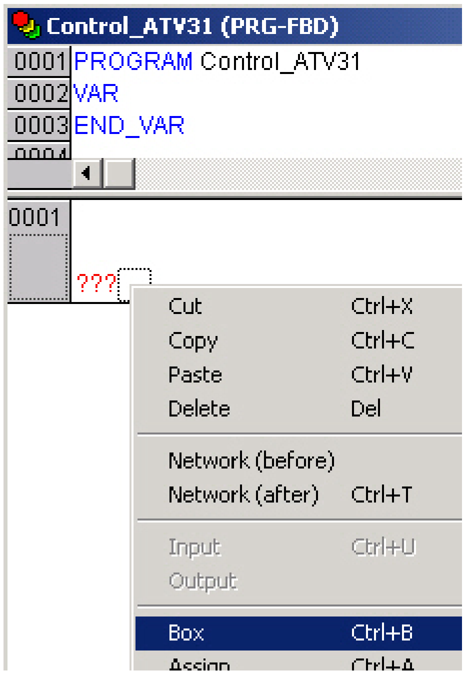
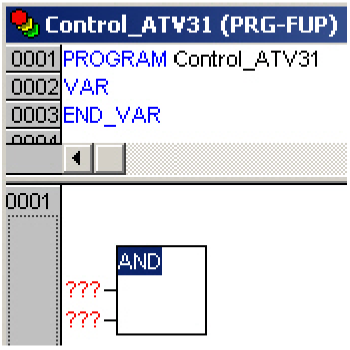
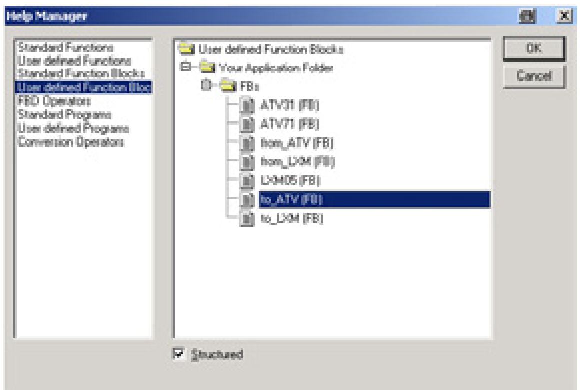
(* ATV Start Operation      ###     step 4   ### *)
IF (ATV_Ready AND Start AND NOT Dir) THEN
     ATV_OUT.data := 16#000F;
ELSIF (ATV_Ready AND Start AND Dir) THEN
     ATV_OUT.data := 16#080F;
END_IF;

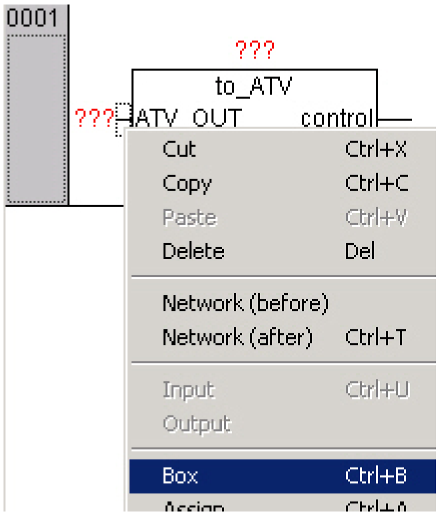
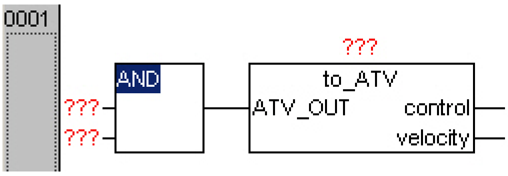
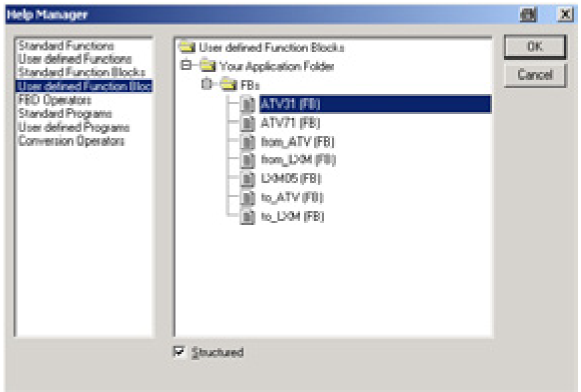
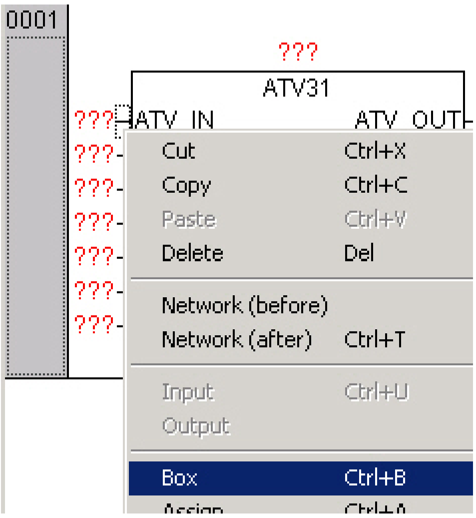
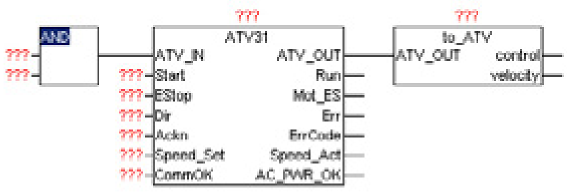
(* ATV in Operation         ###     stay running     ### *)
IF (ATV_Run AND Start AND NOT Dir) THEN
     ATV_OUT.data := 16#000F;
ELSIF (ATV_Run AND Start AND Dir) THEN
     ATV_OUT.data := 16#080F;

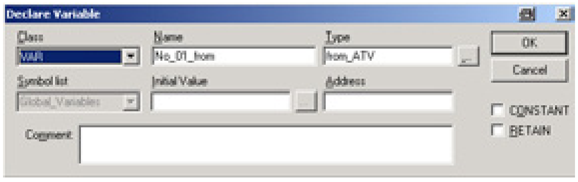
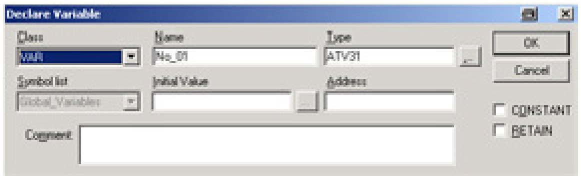
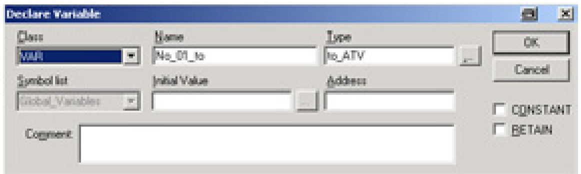
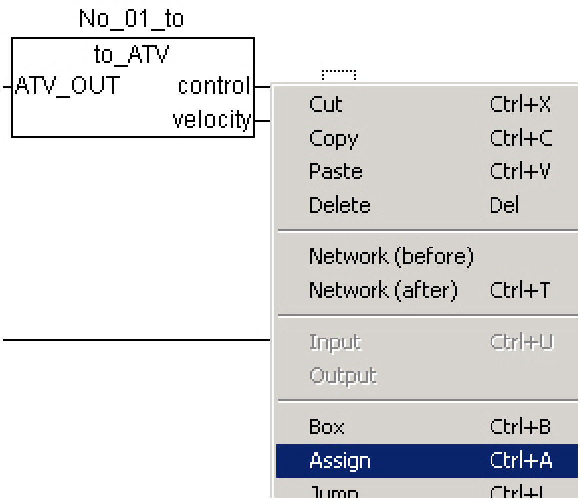
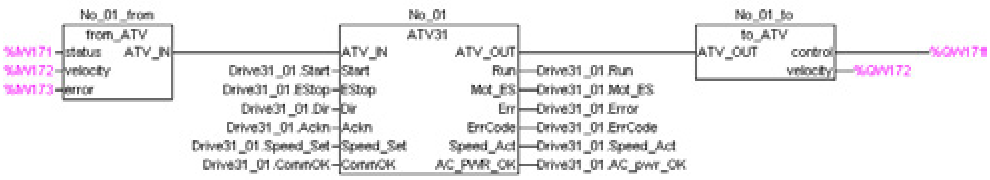
(* ATV Set to Stop          ###     step 5   ### *)
ELSIF (ATV_Run AND NOT Start) THEN
     ATV_OUT.data := 16#0007;
END_IF;
```

Creating a Program Block (FBD) for ATV31

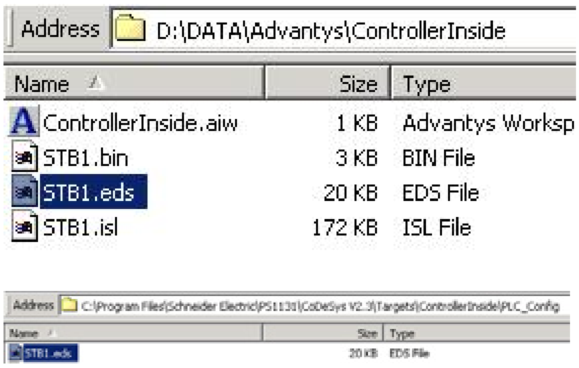
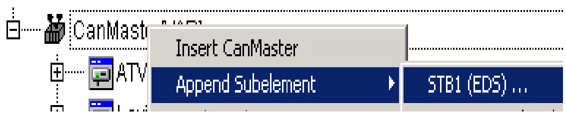
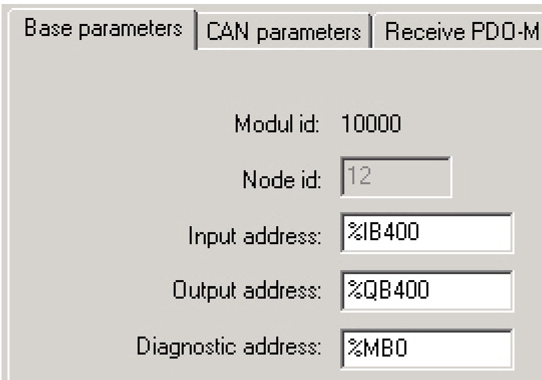
<table>
<tr>
<td>1</td>
<td>Create a program block with the block name.

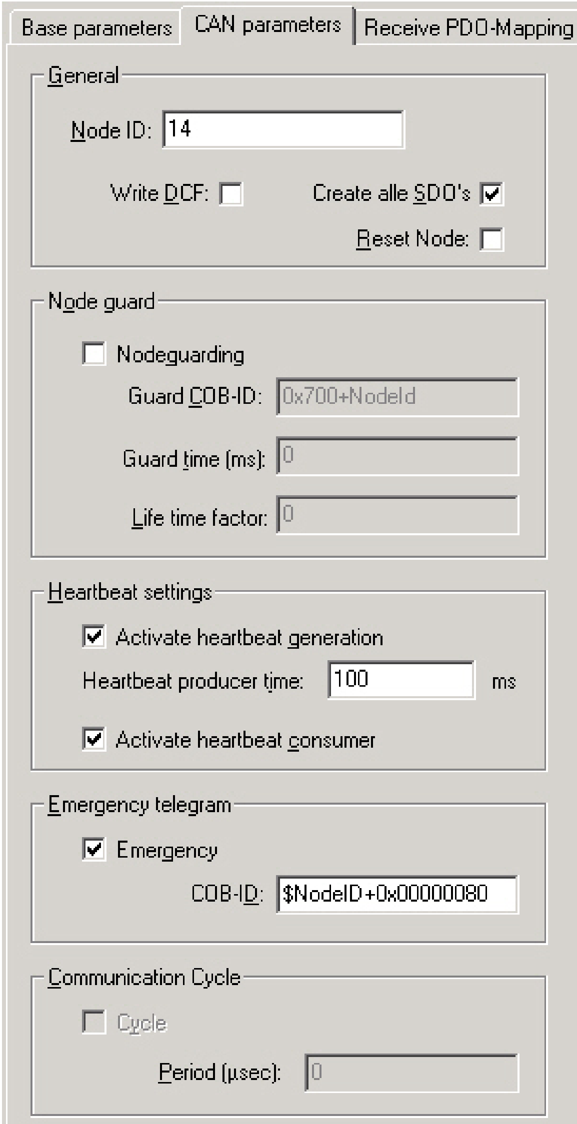
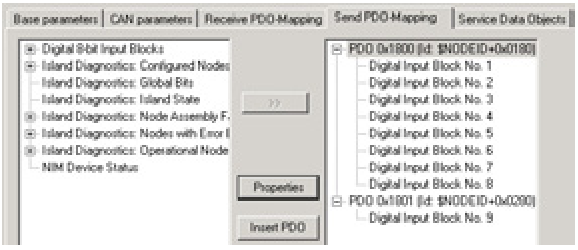
Control_ATV31 in the Your Application Folder.</td>
<td>
</td>
</tr>
<tr>
<td>2</td>
<td>Once the program block has been added, you need to add a block placeholder. Do this by right-clicking with the right mouse button and selecting Block.</td>
<td>
</td>
</tr>
<tr>
<td>3</td>
<td>Press F2 to access the input assistant.</td>
<td>
</td>
</tr>
<tr>
<td>4</td>
<td>Select the to_ATV function block here.</td>
<td>
</td>
</tr>
</table>

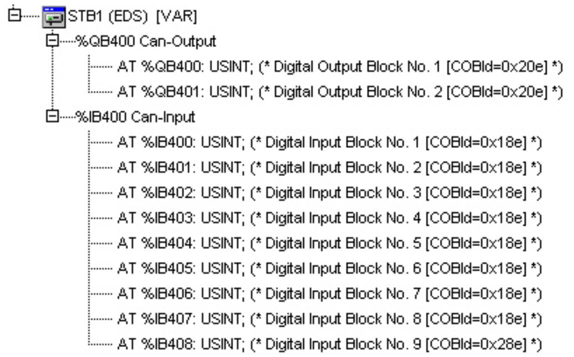
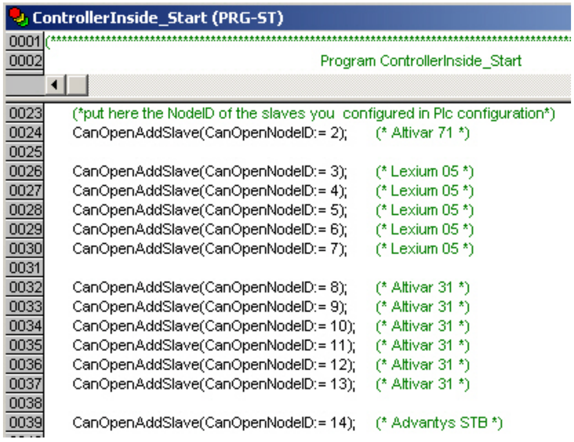
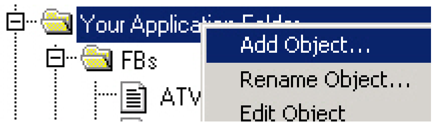
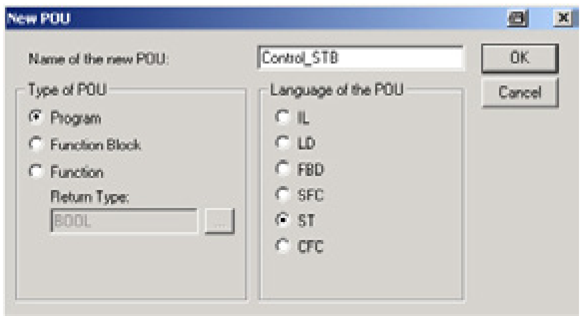
5	Right-click with the mouse to place another **block** at the input pin.	
6	Again, use **F2** to access the input assistant.	
7	And specify FB **ATV31.**	
8	Now place another **block.**	
9	And select the **from_ATV** FB.	

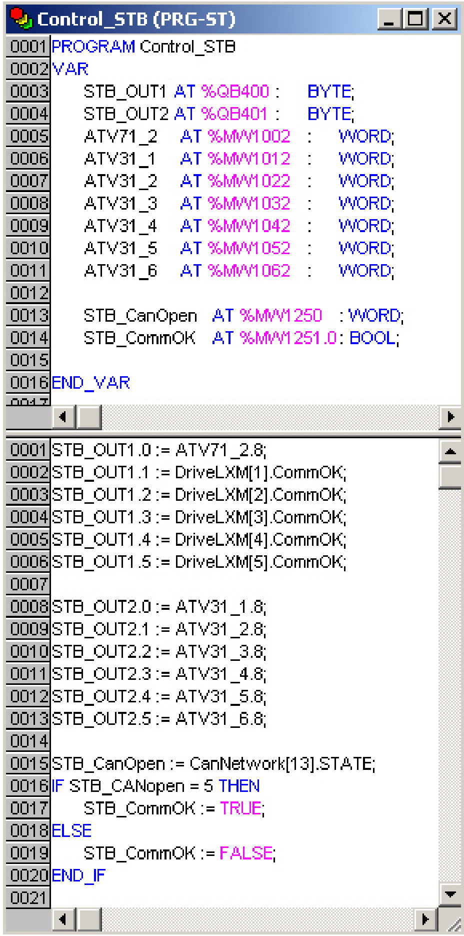
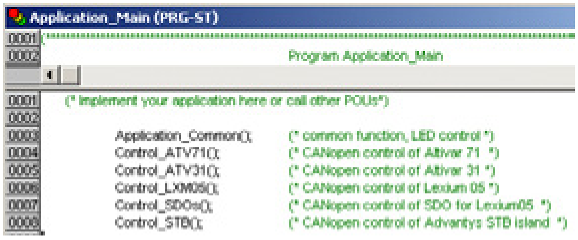
10	Select the instance name (the **???** above the FB) for the FB and enter the instance name **No_01_from**. Once you have confirmed the name, this window will appear. Click **OK** to exit. The entry will be made in the declaration section automatically.	
11	Now do the same for the FB **ATV31** and	
12	and FB **to_ATV** instances.	
13	To connect the output address to the final block, right-click with the mouse and select **Assign**. You will also need to connect the input addresses and variables for the HMI. The figure below shows the first ATV31 fully configured.	
14		

Linking Advantys STB I/O island for CANopen

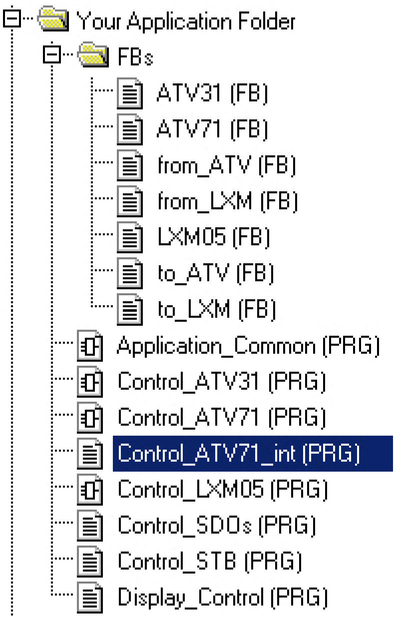
1	Copy the **EDS file** created using the Advantys Configuration Software to the configuration file directory. See also **Integrating the CANopen EDS Files**	
2	Once you have copied the EDS file, right-click with the mouse on the **CanMaster** and select the **STB1** subelement. Alternatively, this function can be accessed via the menu by selecting: **Insert->Append Subelement**	
3	Once you have selected the Advantys STB, a properties dialog box will appear on the right-hand side. On the **Base parameters** tab, enter **%IB400** as the input address and **%QB400** as the output address. (See the list in the Communication chapter for more information.)	

4	In the **CAN parameters** tab: The CANopen address **14** is entered in the Node ID field. **Nodeguarding** is unchecked. **Create alle SDO's, Heartbeat** and **Emergency** are checked. The **Heartbeat producer time** is **100 ms**.	
5	On the **Receive PDO-Mapping** tab (STB <- PLC), the default setting for the first PDO is accepted.	
6	On the **Send PDO-Mapping** tab (STB <- PLC), the default settings for the first two PDOs are accepted.	

7	The **address assignment** appears in the **PLC configuration**.	
8	When the PLC starts up, the **ControllerInside_Start** program is called once and the connected CANopen nodes are declared. The node with address 14 is added here. **CanOpenAddSlave(CanOpenNodeID:= 14);**	
9	To add a new program block, select the **Block** tab in the Object Organizer. Then right-click with the mouse and select: **Add Object….**	
10	Next, assign the **Name of the new POU**, select **Program** and under **Language of the POU**, select **ST** (or another language).	

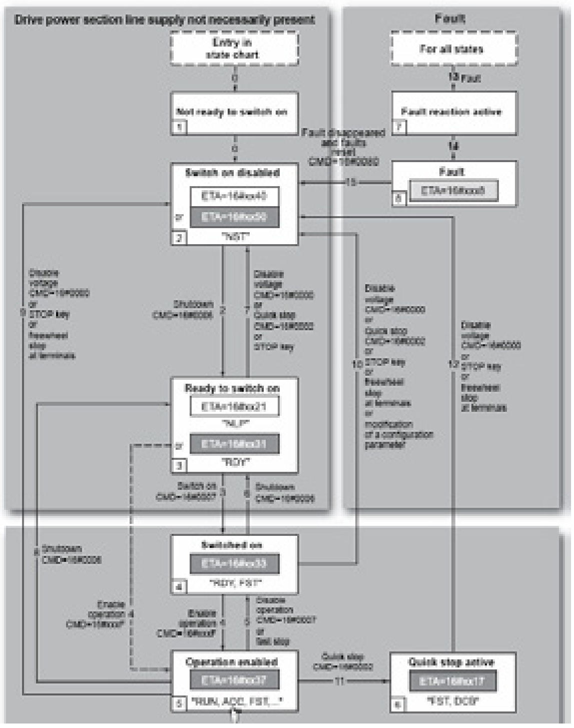
<table>
<tr>
<td>11</td>
<td>This block is used to notify the STB island digital outputs of CANopen bus node availability.</td>
<td>
</td>
</tr>
<tr>
<td>12</td>
<td>In order for the program block to be executed, it must be called by a superordinate program block. This happens here.</td>
<td>
</td>
</tr>
</table>

Altivar 71 for internal data exchange

<table>
<tr><td>1</td><td>The **Control_ATV71_int** program block is responsible for controlling the ATV71 internally.</td><td>
</td></tr>
<tr><td>2</td><td>As with the other drives, the operating state chart is absolutely fundamental to control.

In contrast to the CANopen nodes, the status/actual value and command/setpoint are not transmitted via the bus, but as an internal communication. The special PS1131 software blocks are used for this purpose. As its name suggests, the

MANDATORY_AT_EACH_CYCLE

block, which manages data exchange, is absolutely essential.</td><td>
</td></tr>
<tr><td>3</td><td>The **DrivecomStateGet** block is used to obtain the VSD's status.

Depending on the status, the corresponding variable is set from 1.</td><td>

```
DrivecomStateGet(
    bNotReadyToSwitchOn=> step1,
    bSwitchOnDisabled=> step2,
    bReadyToSwitchOn=> step3,
    bSwitchedOn=> step4,
    bOperationEnabled=> step5,
    bMalfunction=> step8,
    bMalfunctionReactionActive=> step7,
    bQuickStopActive=> step6);
```

</td></tr>
</table>

<table>
<tr>
<td>4</td>
<td>Depending on the HMI setting (and later the application setting), the corresponding blocks for controlling the VSD are called.

The following blocks are used:

• DriveRunForward()
• DriveRunReverse()
• DriveStopRamp()
• DriveStopQuick()
• DriveStopFreeWheel()
• DriveFaultReset()

For the setpoint and actual value:

• DriveTargetVelocitySet()
• DriveActualVelocityGet()

For reading out the error code:

• DriveParameterRead1()

As part of this process, the content of address 16#219E is scanned.</td>
<td><pre>
IF Drive_01.Start AND NOT Drive_01.Dir THEN
    DriveRunForward();
END_IF;

IF Drive_01.Start AND Drive_01.Dir THEN
    DriveRunReverse();
END_IF;

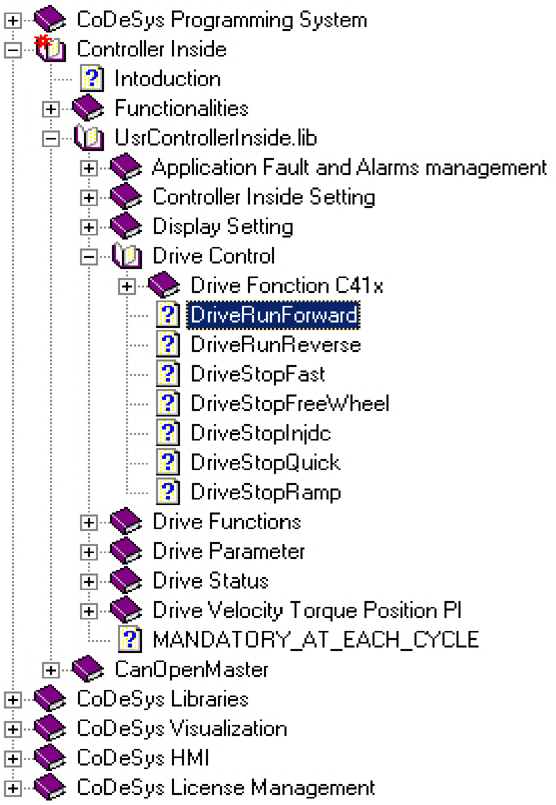
IF NOT Drive_01.Start THEN
    DriveStopRamp();
END_IF;

IF Drive_01.EStop THEN
    Drive_01.Start := FALSE;
    DriveStopQuick();
END_IF;

IF stop THEN
    Drive_01.Start := FALSE;
    DriveStopFreeWheel();
END_IF;

IF Drive_01.Ackn THEN
    reset := TRUE;
ELSE
    reset :=FALSE;
END_IF;
    DriveFaultReset(bStatus=> reset);

IF step7 OR step8 THEN
    Drive_01.Start := FALSE;

END_IF;
</pre></td>
</tr>
<tr>
<td>5</td>
<td>A description of the blocks is available in the online help.</td>
<td>
</td>
</tr>
</table>

Using the plug-in graphic display terminal

<table>
<tr><td>1</td><td>The drive has 50 parameters which are also referred to as display parameters. These can be viewed on the graphic display terminal. The display parameters function as an exchange table between the display and Controller Inside PLC. The variable names are Display_Oxx (xx=1...50) and are listed in the PLC configuration under

Local Drive -> Display Parameters.

The PLC reads and writes the display parameters automatically, assuming that they have been configured. Writing of the parameters takes place in conjunction with the AppliLock parameter element.</td><td>Controller Inside
Digitals Inputs[FIX]
Digitals Outputs[FIX]
Analogs Inputs[FIX]
Analogs outputs[FIX]
Local Drive[FIX]
Drive Cyclic Parameters Read[FIX]
Drive Cyclic Parameters Write[FIX]
Drive IOs[FIX]
EMPTY-SLOT[SLOT]
Display parameters[FIX]
Display_O01 AT %QW30: WORD;
Display_O02 AT %QW31: WORD;
Display_O03 AT %QW32: WORD;
Display_O04 AT %QW33: WORD;
Display_O05 AT %QW34: WORD;</td></tr>
<tr><td>2</td><td>When the application is started, the ControllerInside_Start program block is called via PLC_PRG -> Drive_Start. Steps 1 to 5 are executed.</td><td>1: Display_SetupList();
IF Display_SetupList.bDone THEN StateConfigure:=2; END_IF

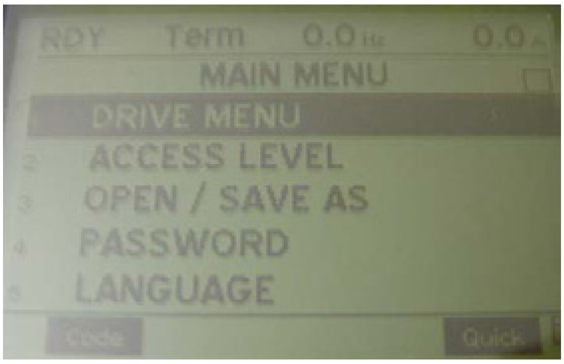
2: Display_SetupParameter();
IF Display_SetupParameter.bDone THEN StateConfigure:=3; END_IF

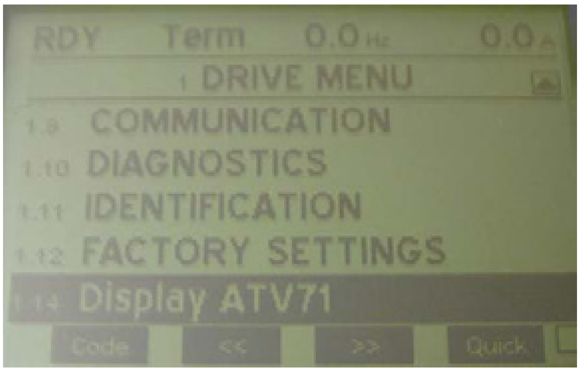
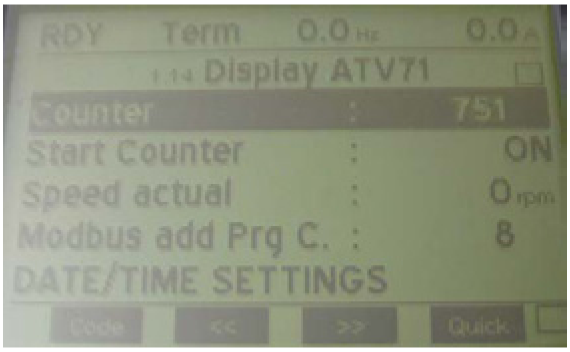
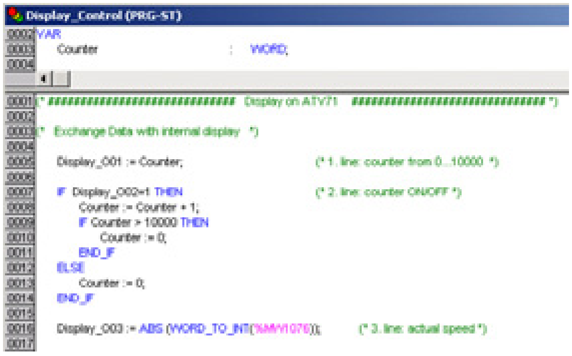
3: DisplayMenuTextSet(TextmenuLong:= 'C Inside', TextmenuShort:='Cins');
StateConfigure:=4;

4: Display_SetupText();
IF Display_SetupText.bDone THEN StateConfigure:=5; END_IF

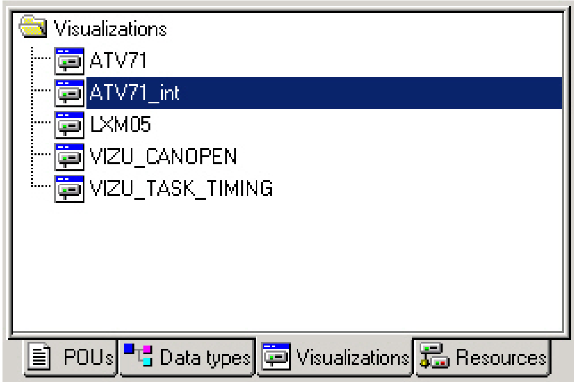
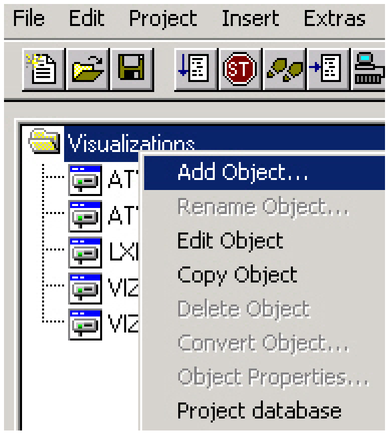
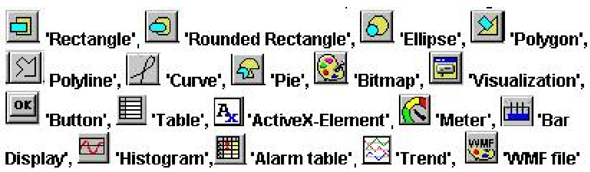
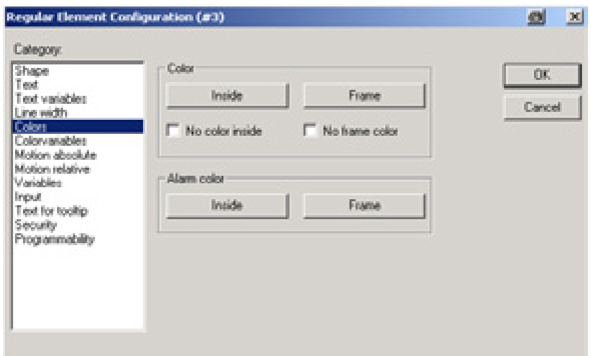
5: Display_RestoreSavedParameters();
IF Display_RestoreSavedParameters.bDone THEN StateConfigure:=1; bDone:=TRUE; END_IF</td></tr>
<tr><td>3</td><td>The second step defines whether a variable is visible and whether it can be modified. The limits and display format are also defined.</td><td>(* Configure the Display parameters*)
DisplayNumericSet(Indice:=1 , (* Display parameter to be configured 1 TO 50 *)
Visibility:=TRUE , (* TRUE : Display it ; FALSE : Hidden *)
AppliLock:=TRUE, (* TRUE : Only Controller Inside can write it *)
RunLock:= FALSE, (* TRUE : No write access when drive is RUN *)
Save:=FALSE, (*Save the value in NVRAM and Restore it on Power up *)
Signed:= FALSE, (* TRUE : the Display parameter is signed *)
Minimum:= 0, (* minimum *)
Maximum:= 10000, (* maximum *)
Default:=0 , (* default value when Factory setting selected *)
Decimal:=0); (* decimal ; 1: 0.1 ; 2: 0.01 ; 3:0.001 *)

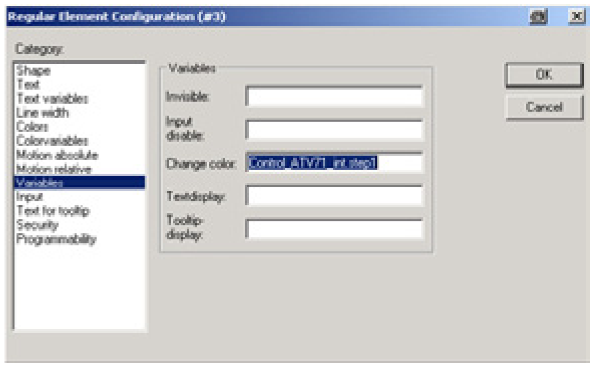
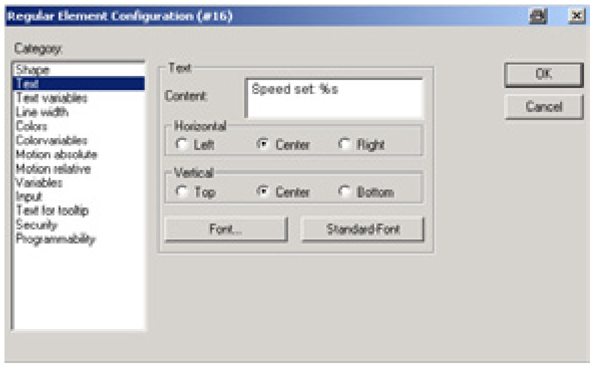
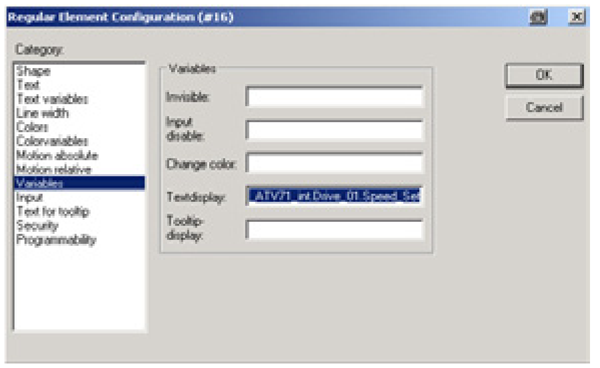
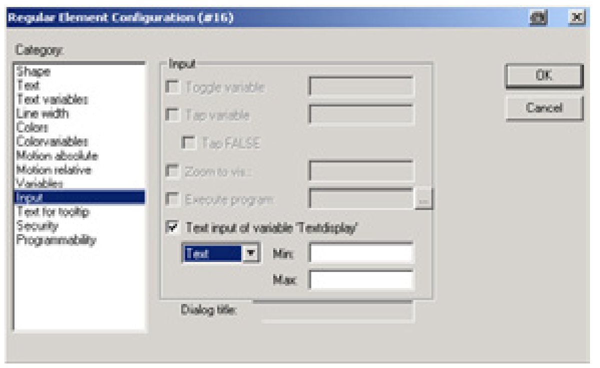
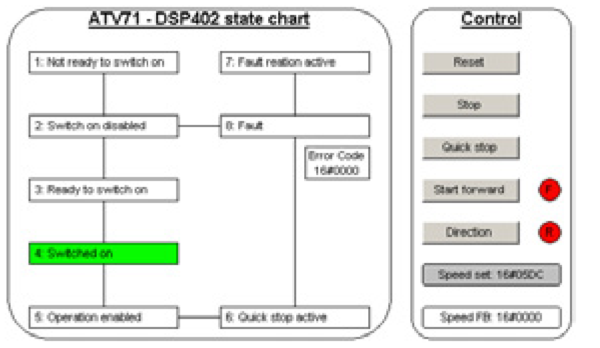
DisplayList_ON_OFF_Set(Indice:=2 ,
AppliLock:=FALSE , RunLock:= FALSE, Save:= FALSE, Default:= 0);

DisplayNumericSet(Indice:=3 ,
Visibility:=TRUE ,AppliLock:=TRUE ,RunLock:=FALSE ,Signed:=TRUE ,Minimum:=0 ,Maximum:=150C</td></tr>
<tr><td>4</td><td>As part of the third step, the name of the display that is to appear under Point 1.14 and in the header is parameterized. Enter Display ATV71 here.</td><td>3: DisplayMenuTextSet(TextmenuLong:= 'Display ATV71', TextmenuShort:='DSP');
StateConfigure:=4;</td></tr>
<tr><td>5</td><td>The display can be found in the menu under 1. Drive Menu</td><td></td></tr>
</table>

<table>
<tr><td>6</td><td>and
1.14 Display ATV71.</td><td>
</td></tr>
<tr><td>7</td><td>The display texts are defined as part of the fourth step.</td><td>DisplayTextSet(Indice:=1, TextParameter:= 'Counter' , TextUnit:='');
DisplayTextSet(Indice:=2, TextParameter:= 'Start Counter ' , TextUnit:=' ');
DisplayTextSet(Indice:=3, TextParameter:= 'Speed actual' , TextUnit:='rpm ');
(*DisplayTextSet(Indice:=4, TextParameter:= ' ' , TextUnit:=' '); *)</td></tr>
<tr><td>8</td><td>Study the example.
A counter is started in the PLC via Start Counter and the value (0 to10000) is displayed here under Counter.
Also, the current speed, in RPM, is displayed next to Speed actual.</td><td>
</td></tr>
<tr><td>9</td><td>The logic is programmed in the Display_Control program block.</td><td>
</td></tr>
</table>

Creating viewer within CoDeSys

1	The PS1131 (CoDeSys) programming software features an integrated viewer. If the PLC has been programmed with CoDeSys, no additional tools are required to visualize the PLC data, which means you already have everything you need for monitoring and operation. The programming system contains an integrated viewer/editor, which enables users who are involved in application development to generate display screens without having to switch interfaces.	
2	A visualization object is a project block that is managed via the "Visualizations" tab. It contains an array of visualization elements and can have certain properties assigned to it as the global object. One or more visualization objects can be created within the context of a project and they can even be linked to one another.	
3	To create a visualization object in the Object Organizer. Select the **Visualizations** folder followed by **Add Object...** This will open the dialog box for new visualization objects, where you should enter the name of the new visualization object.	
4	A visualization element is a graphical element, which is used as part of the process of creating a visualization object. The potential elements are made available via a menu bar. Each element is configured separately.	
5	Here you can define the normal color and the alarm color in order to facilitate a subsequent change in color.	

6	The change in color is achieved by entering the variables. Press **F2** to access the input assistant.	
7	In order for analog values to be displayed, enter the **%s** placeholder in the text content field.	
8	When defining the variables, enter the variable to be displayed in the Textdisplay field. Use **F2.**	
9	In order to be able to modify the variable within the context of visualization (as well as viewing it), you must select **Text input of the variable** here. Alternatively, binary variables can also be switched.	
10	This example depicts animated visualization **ATV71_int.** You will find that the startup project already contains two instances of visualization. The first of these (**VIZU_TASK_TIMING**) indicates the cycle time states. The other one (**VIZU_CANOPEN**) indicates the state of the CANopen bus.	

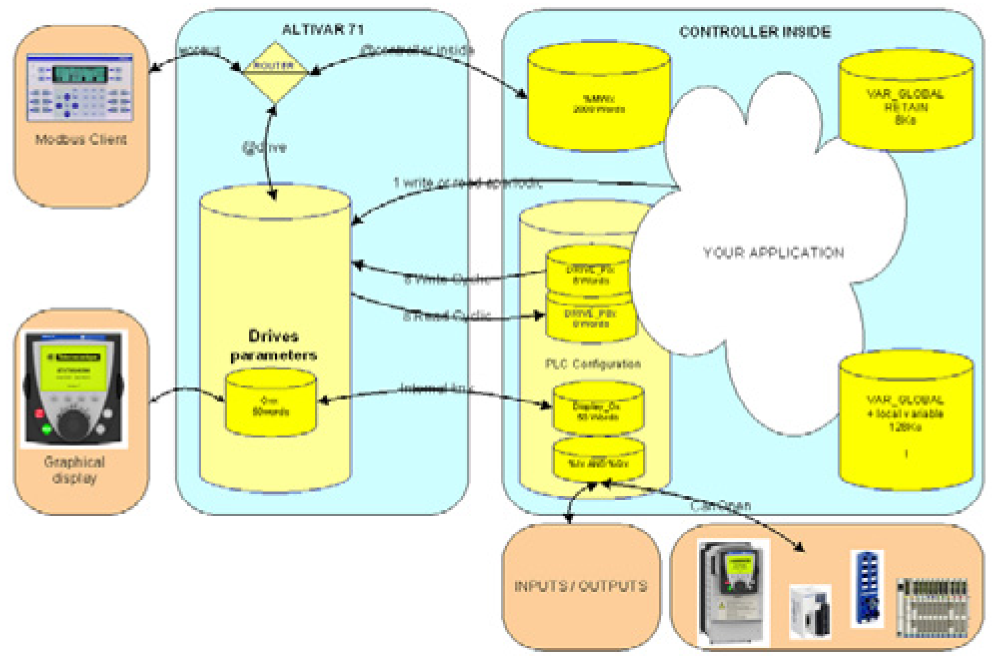
Data exchange with an external HMI

There are **2000 words** available for data exchange via Modbus.

It is important to note that the Modbus port on the Altivar 71 has **two Modbus** addresses. One is used to access the Altivar71 and the other to access the Controller Inside card.

The addresses can be entered via the display or via PowerSuite.

For a list of variables used, see Communication. This list also provides details of the %MW addresses and content.



Devices

Introduction

This chapter describes the steps required to initialize and configure the devices to attain the described system function.

General

Descriptions are provided in respect of the following devices:

- I/O Platform - Advantys STB
  The Advantys Configuration Software is required to load the configuration onto the island and also to generate the EDS file for the PLC.

- Altivar31, Altivar 71 and Lexium05
  The Altivar VSD settings can also be entered or modified via the front panel. You also have the option of using the PowerSuite software. The advantages of using PowerSuite are that you

  - Can save the data on your PC and copy it as you wish
  - Can print out the documentation *and*
  - Can be assisted in optimizing the parameters online.

I/O Platform - Advantys STB

Introduction

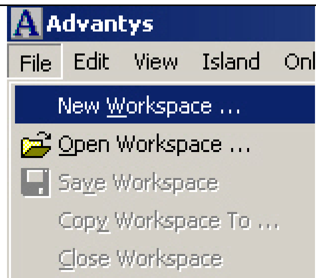
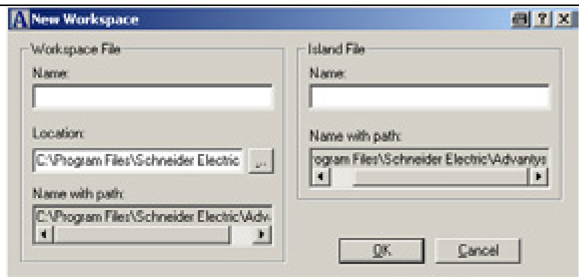
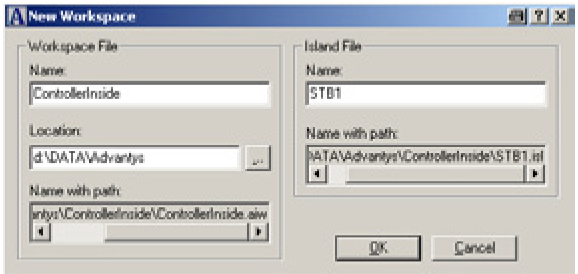
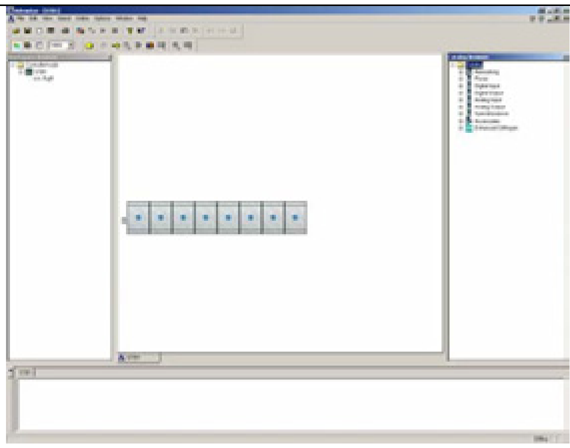
This chapter describes how the Advantys I/O platform is configured using the Advantys Configuration Software.

Proceed as follows:

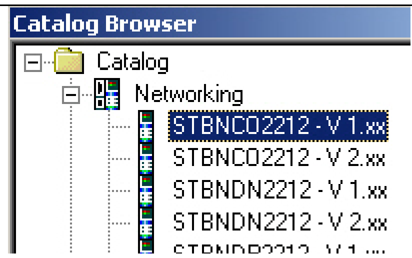
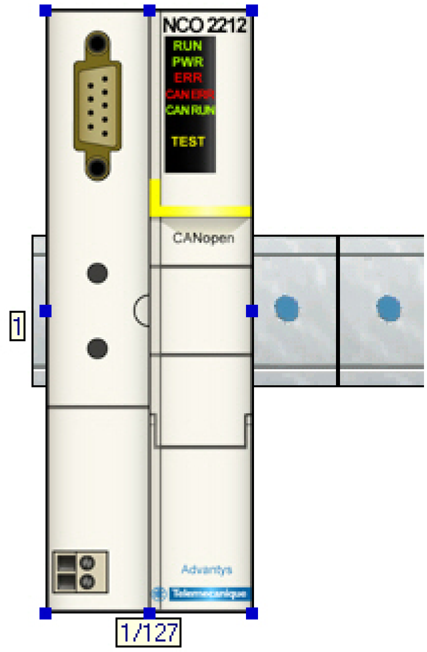
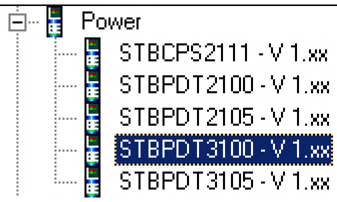
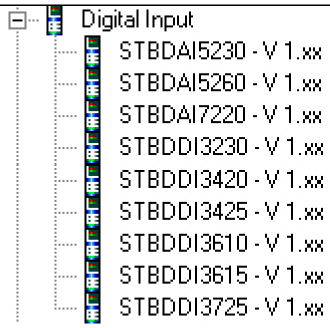
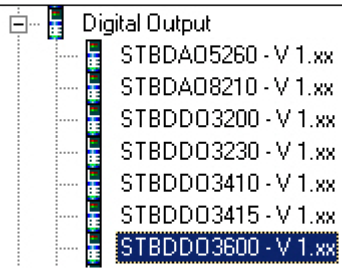
- **Create a new project (workspace)**
- **Configure the hardware (network interface, power supply and I/O modules)**
- **Configure CANopen extension communication (baud rate)**
- **Download configuration to island**
- **Generate EDS file**

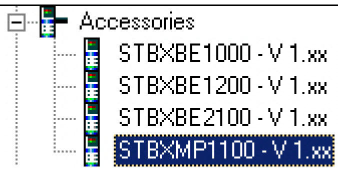
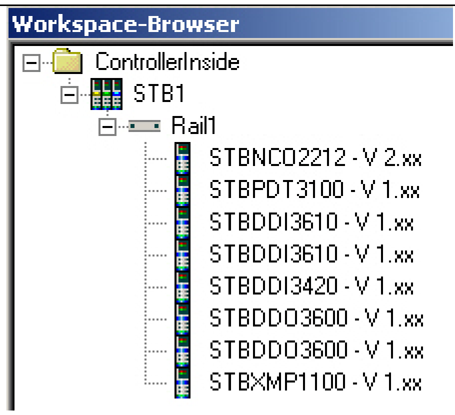
Creating a New Project (Workspace)

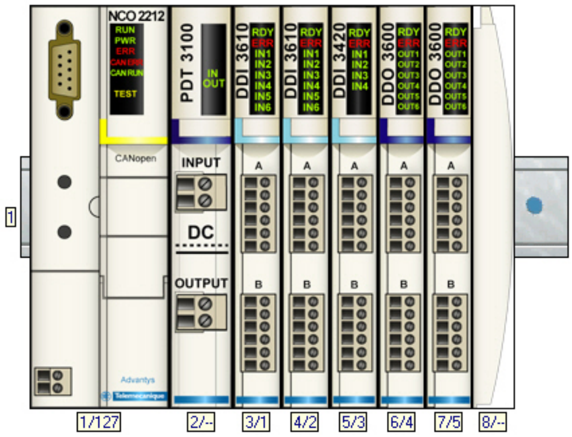
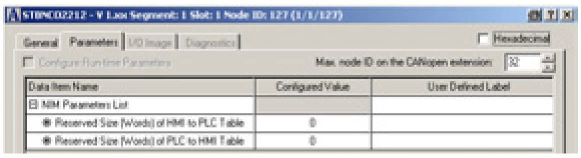
1	Once you have installed and started the Advantys Configuration Software, you will be presented with a choice between **Advantys STB** Advantys FTB, FTM, and OTB. Select the first of these options.	

2	Then select the language.	
3	After starting the Advantys configuration software, you must create a new workspace.	
4	To do this, specify the **path**, the **workspace name** and the **name** of the first **island**.	 
5	An empty top-hat rail will be displayed.	

Configuring the hardware

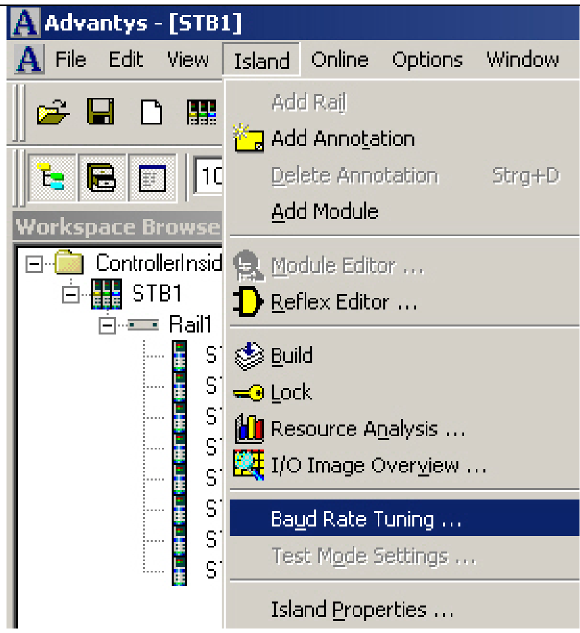
1	Then select the network interface for CANopen. **STB NCO 2212**	 
2	Next, select the power supply **STB PDT 3100,**	
3	the digital input cards used **STB DDI 3610** (2x), **STB DDI 3420** (1x),	
4	the digital output cards **STB DDO 3600** (2x)	

<table>
<tr><td>5</td><td>and finally the terminating resistor
STB XMP 1100.</td><td>
</td></tr>
<tr><td>6</td><td>The following should now be displayed.</td><td>


</td></tr>
<tr><td>7</td><td>Double-click on the CANopen header to display the Properties page.
Here you can enter a size for your exchange table on the Parameters tab. Our example does not involve the use of an exchange table so the value is set to 0.</td><td></td></tr>
</table>

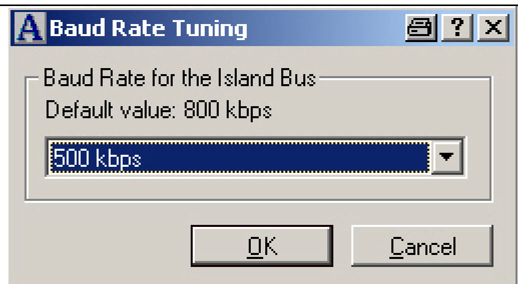
Configuring internal CANopen bus communi-cation (baud rate)

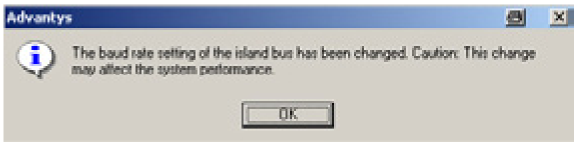
1 The internal baud rate can be set via the menu bar by selecting **Baud Rate Tuning….**



2 The rate used is **500 kbps.**

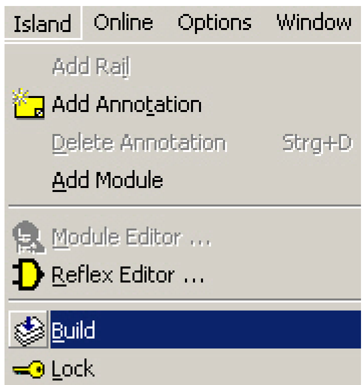
Note:
Set the parameter for the transfer rate between NIM and PLC with the two rotary switches on the front of the NIM. See Communication for further details.



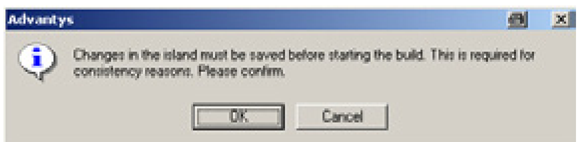


Downloading configuration to island

1 Parameterization is now complete. Now you need to generate the STB project. To do this select **Island -> Build** from the menu bar.

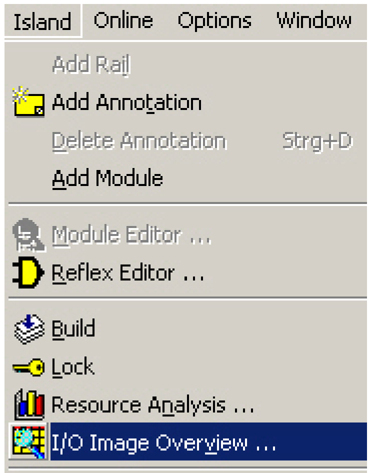


2 Unless you have already saved the changes, you should do this now by clicking **OK.**

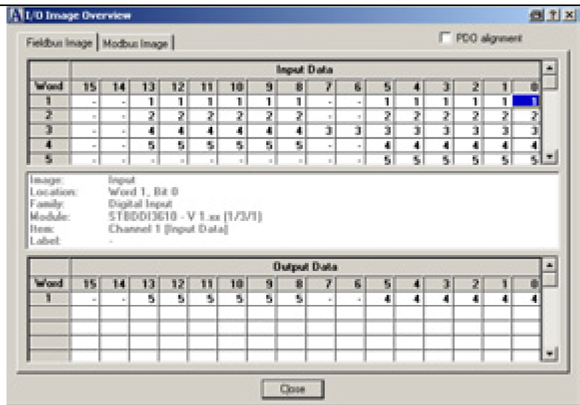


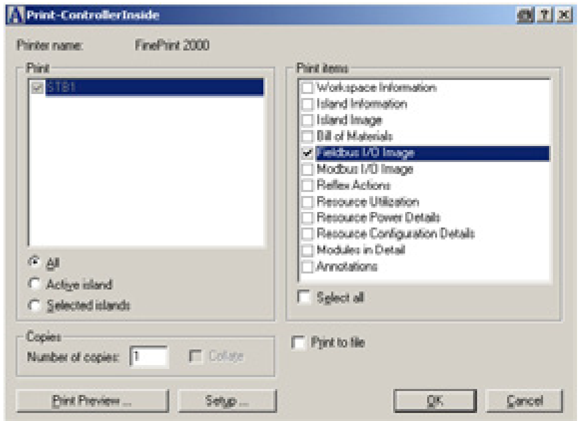
<table>
<tr><td>3</td><td>The bottom section provides a log of the individual actions.</td><td></td></tr>
<tr><td>4</td><td>The words

... completed successfully

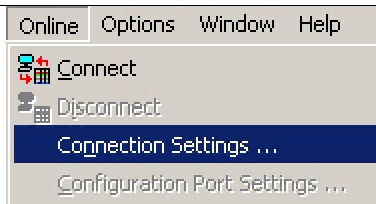
should now be visible here.</td><td>Saving island file ...
Island file has been saved.
Build of island data in progress ...
Build completed successfully.</td></tr>
<tr><td>5</td><td>To find out how the individual inputs and outputs fit into the data exchange process, you can call the I/O Image Overview.</td><td>
</td></tr>
</table>

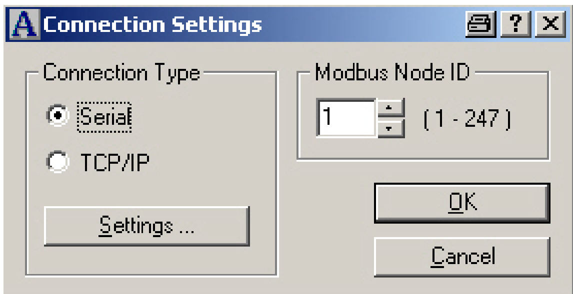
<table>
<tr>
<td>6</td>
<td>You must ensure that the Fieldbus Image tab is selected.
Select an input or output word element here to display its content in the intermediate window.

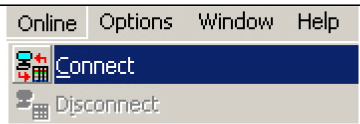
Note:
Alternatively, this information can also be printed out. To do this, select Fieldbus Image.</td>
<td>


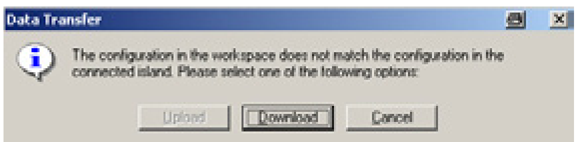


</td>
</tr>
<tr>
<td>7</td>
<td>Before you can load the configuration, you must first define the appropriate settings under Online -> Connection Settings.

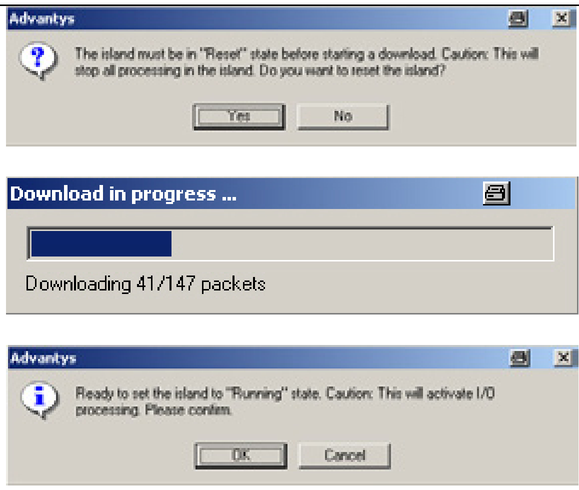
As the serial cable is being used, select the following:
  Serial
  Modbus Node ID: 1</td>
<td>




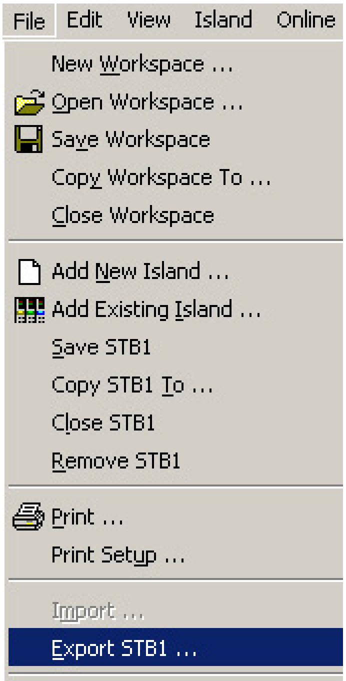
</td>
</tr>
<tr>
<td>8</td>
<td>Establish a connection via Online -> Connect. The NIM and PC configuration versions will now be compared. If they do not match, this window (see right) will appear. Select Download.</td>
<td>


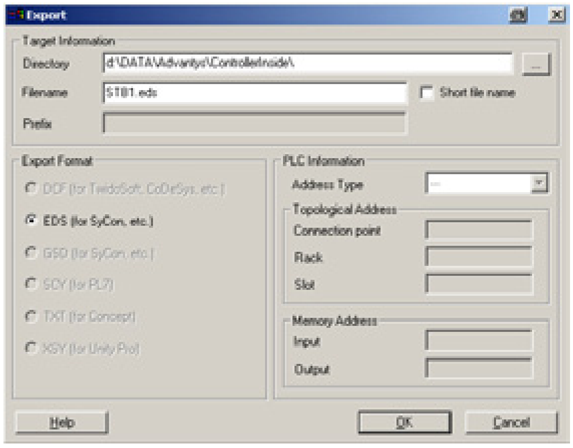
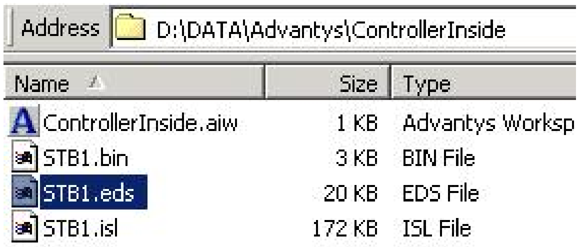
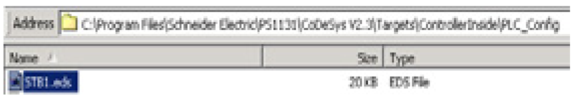


</td>
</tr>
</table>

9	Then respond to the subsequent messages with **Yes** And **OK**.	
10	The text shown on the right should now appear in the bottom window.	**Island is healthy.**
11	Then, you can terminate the connection.	

Generating an EDS file

1	To generate the EDS file, select **File -> Export STB1….**	

<table>
<tr><td>2</td><td>You can enter the directory and file name here.</td><td>
</td></tr>
<tr><td>3</td><td>Copy this file from …</td><td>
</td></tr>
<tr><td>4</td><td>to the PS1131 (CoDeSys) directory.</td><td>
</td></tr>
</table>

PowerSuite

Introduction

PowerSuite is a tool for configuring and supervising motor controllers. Using Powersuite, the user can define machines, their configuration and the communications parameters.

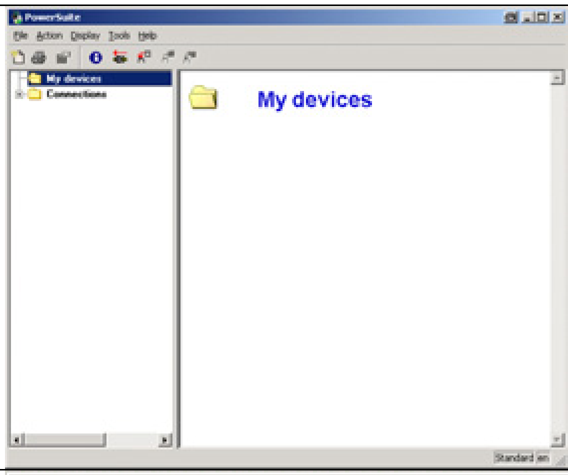
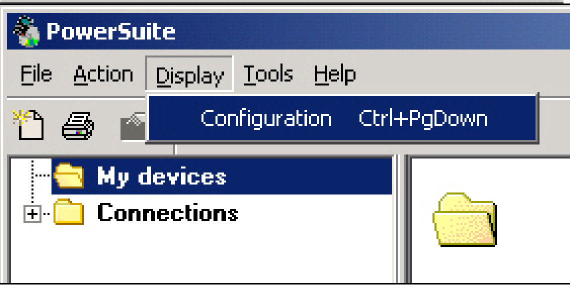
Powersuite offers the advantage that you can

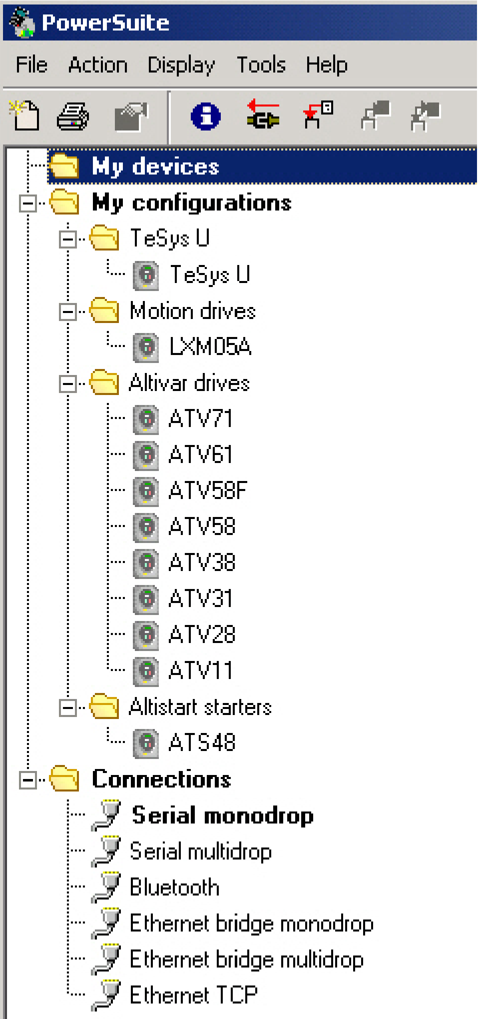
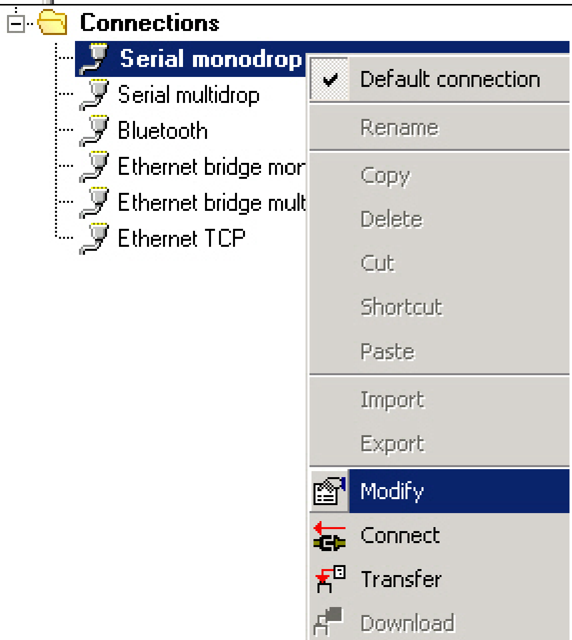
- Save data to your hard drive and duplicate it
- Print documentation for your project
- Optimise your parameters online

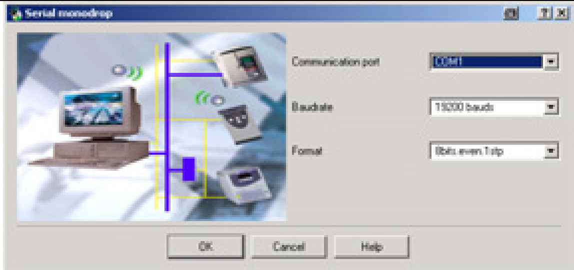
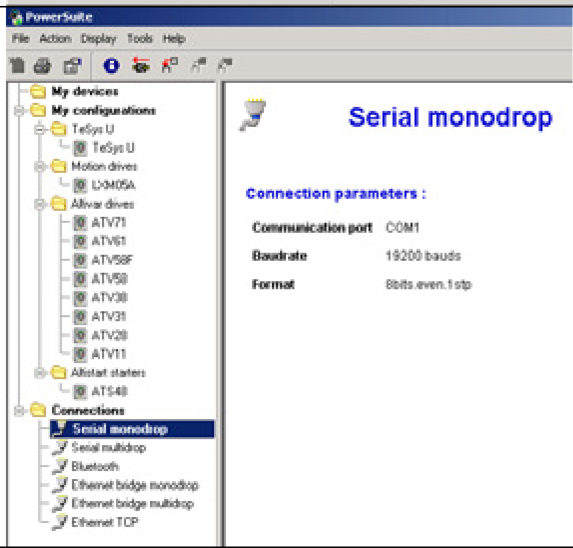
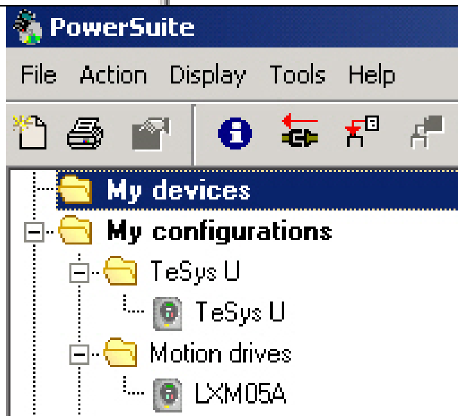
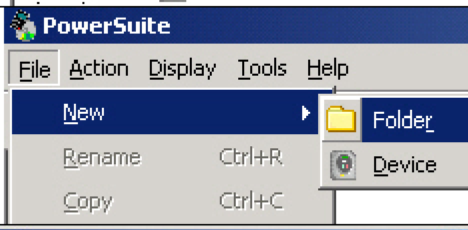
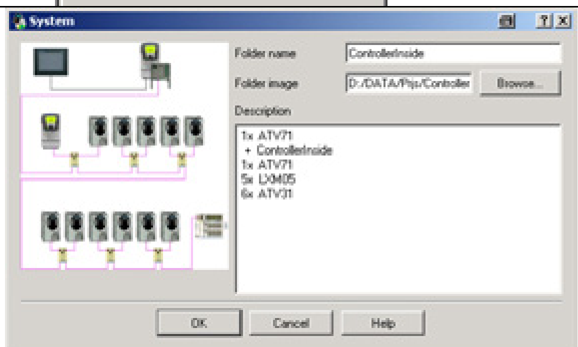
The version described here can be used for all controllers used in this configuration (ATV31,ATV71 and LXM05). Each of the mentioned products also has its own ‚light' version, but these versions can only be used for the particular product they accompany.
.

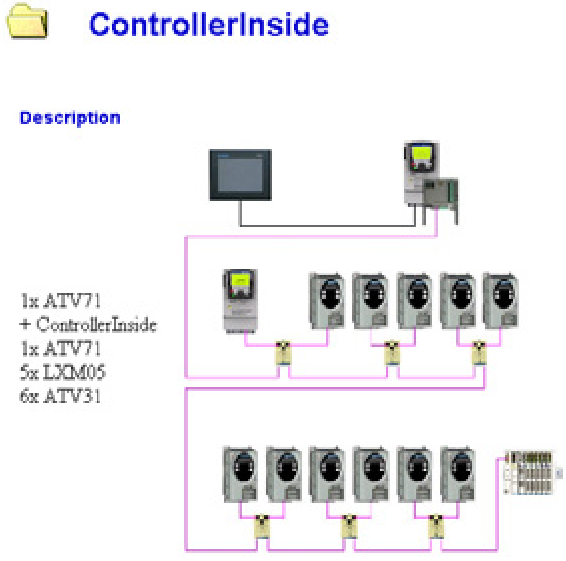
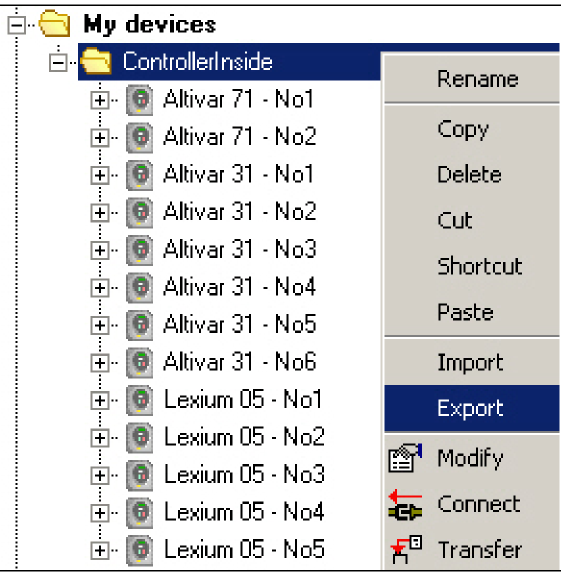
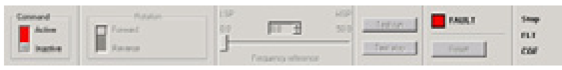
General Setup

The following describes the basic setup of PowerSuite:

1	After starting PowerSuite you see the display on the right.	
2	Select **Display->configuration** so that the configuration is added to the project browser list.	

3	The project browser aids you in managing the machines.	
4	In **Connections** You can view the communications setup and **Modify** it.	

5	Select the **COM** interface you wish to use .	
6	When you select the connection in the project browser, the description appears in the window on the right.	
7	Powersuite allows you to create folders for your drives. To do this click on the main folder: **My Devices**	
8	Then select: **File->New->Folder**	
9	In the dialog that opens, input the **Folder name**. You can also add a **Folder image** and a **Description**. Exit with **OK** to add the folder to the project browser	

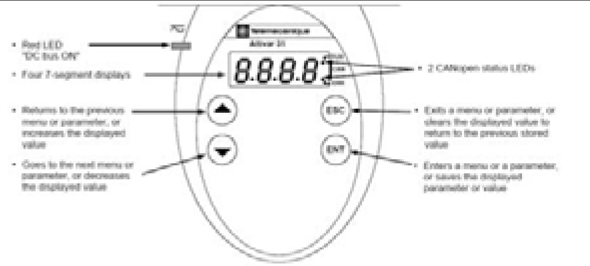
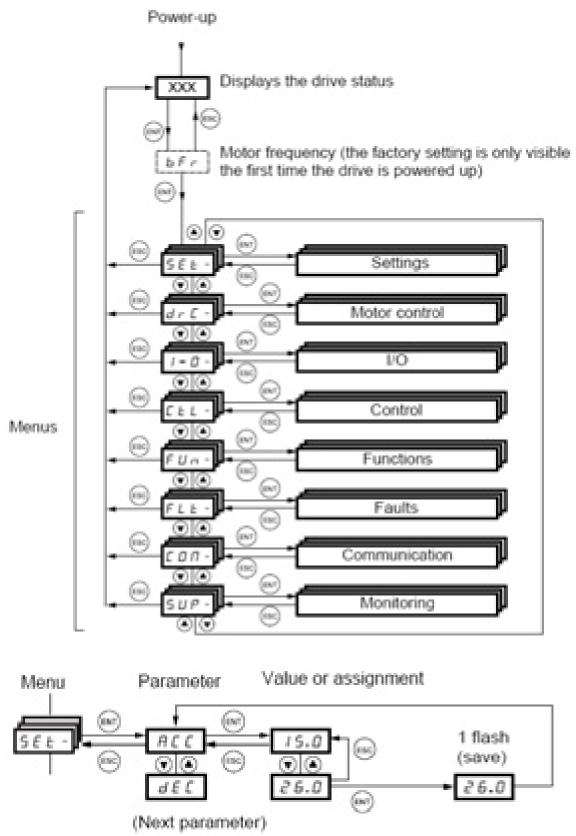
<table>
<tr><td>10</td><td>If you select your folder in the project browser your image and description are shown in the window on the right.</td><td>
</td></tr>
<tr><td>11</td><td>Any drive group, including its data, can be exported.</td><td>
</td></tr>
<tr><td>12</td><td>Once PowerSuite is connected to the drive you can used the control panel to operate the drive.</td><td></td></tr>
</table>

Introduction

The settings for the ATV31 variable speed drive can either be made manually using the control panel on the device or by means of the PowerSuite configuration software.

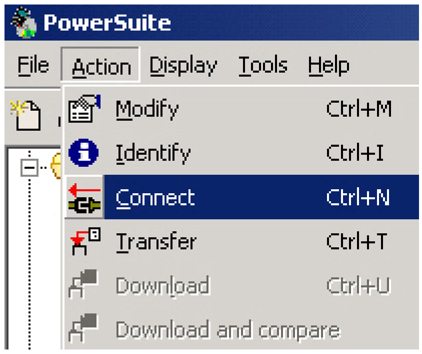
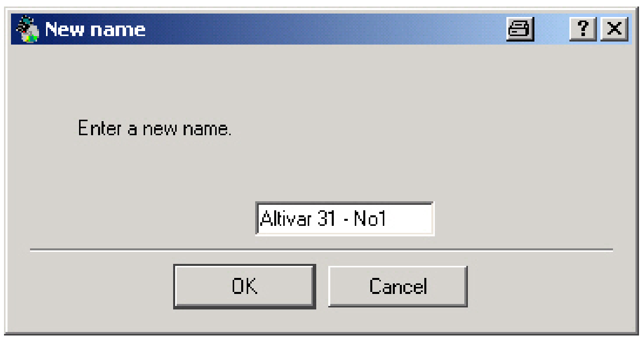
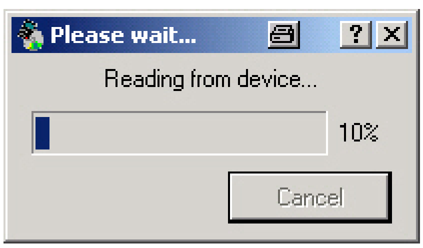
Configuring ATV31 with Control Panel

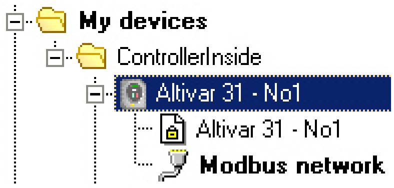
The ATV31 parameters can also be entered or modified via the control panel on the front of the device. This section describes how to set the drive using this control panel.

1	The CANopen address and transfer rate are set manually via the **control buttons** on the device.	
2	First, use the control buttons to select the **Communication** submenu.	
3	In the **Communication** menu, the **CANopen address** must be set in the **AdC0** parameter. In the example software provided, the values **8 to 13** have been set aside for the six VSDs.	
4	In the **Communication** menu, you must also set the transfer rate in the **BdC0** parameter to the value **500.0** (**kbits**).	
5	Alternatively, the address and transfer rate can also be parameterized using the PowerSuite configuration software.	

PowerSuite with ATV31

The parameters can also be set using the PowerSuite configuration software. This section describes the PowerSuite V2.3 software.

1	On startup, select the folder **ControllerInside** in the project browser. (see the general PowerSuite setup above)	
2	Use the **Connect** option in the **Action** menu or the equivalent icon in the tool bar to establish a connection with the device. **Note:** Prior to this, you must have plugged in the connection cable from the PC to the ATV.	
3	PowerSuite notes that it is a new device and asks for permission to insert it in the database Click on **Create**	
4	Enter a name for the new device. This name is used for the configuration.	
5	The data is read out from the ATV31.	

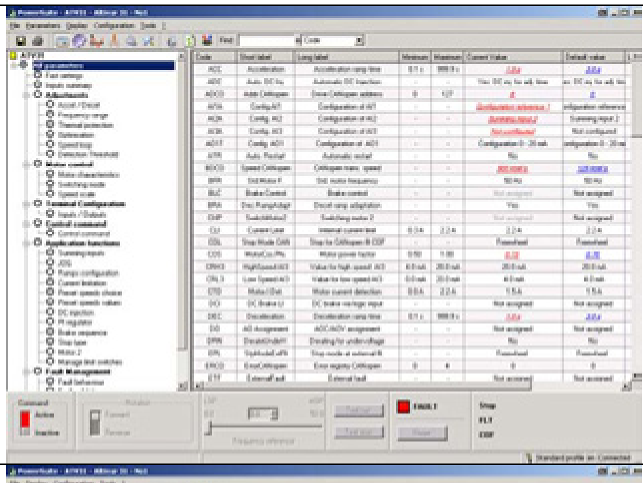
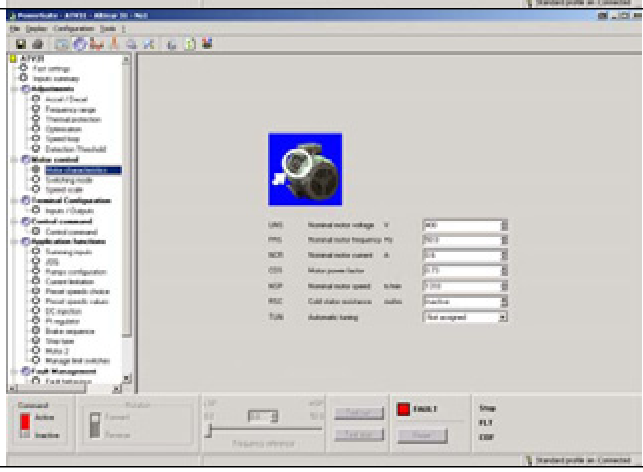
<table>
<tr>
<td>6</td>
<td>Once the transfer is complete, the device data will be displayed.</td>
<td>


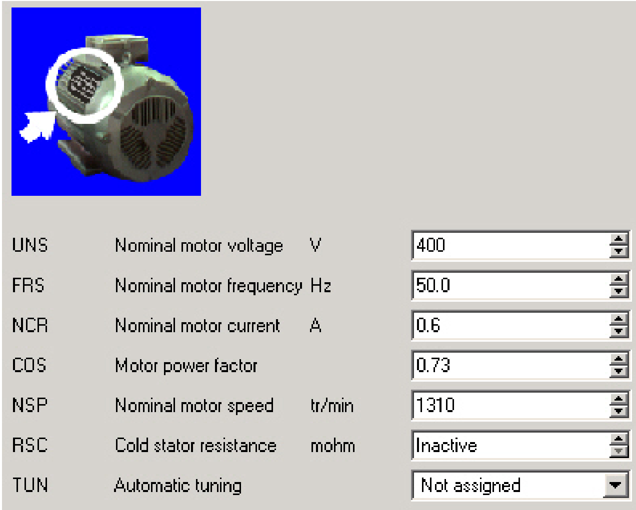
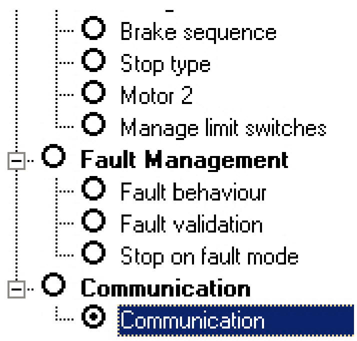
<h2>Altivar 31 - No1</h2>

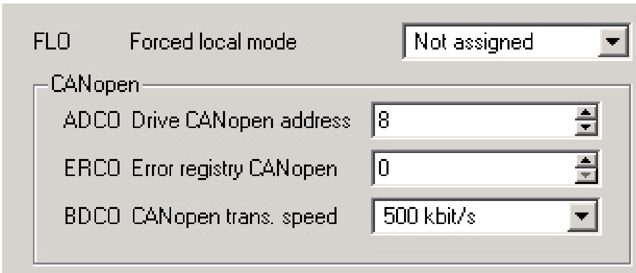
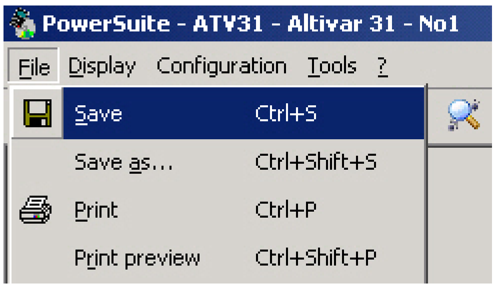
<h3>Characteristics</h3>

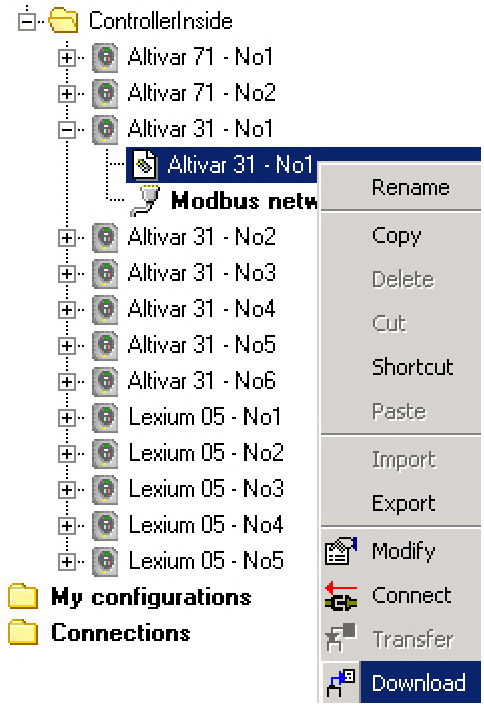
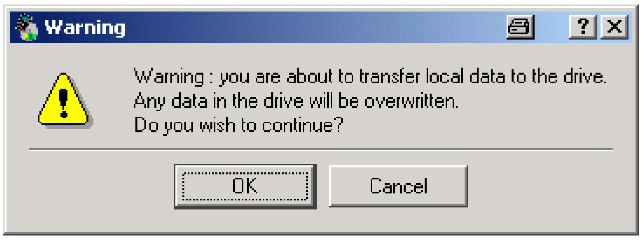
<table>
<tr><td>Reference</td><td>ATV31H037N4</td></tr>
<tr><td>Power</td><td>0.37 kW / 0.5 HP</td></tr>
<tr><td>Voltage</td><td>380/500V three phase</td></tr>
<tr><td>Zone</td><td>Europe</td></tr>
<tr><td>Hardware type</td><td>Product on heatsink</td></tr>
<tr><td>Nominal current</td><td>1,5 A</td></tr>
<tr><td>Max. transient current</td><td>2,3 A</td></tr>
</table>

<h3>Structure</h3>

<table>
<tr><th>Card</th><th>Reference</th><th>Serial number</th><th>Version</th><th>Vendor name</th></tr>
<tr><td>Device</td><td>ATV31H037N4</td><td>XX X5 11 021 076</td><td>V1.2IE03</td><td>TELEMECANIQUE</td></tr>
<tr><td>Control Board</td><td></td><td>XX Xf ff ff ff</td><td>V1.2IE03</td><td>TELEMECANIQUE</td></tr>
<tr><td>HMI Board</td><td>VX4A311/312(A)</td><td>XX Xf ff ff ff</td><td>V1.1IE02</td><td>TELEMECANIQUE</td></tr>
</table>
</td>
</tr>
<tr>
<td>7</td>
<td>The parameters can be displayed and modified in list format</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>or using graphical support. The view shown here can be accessed via:
Settings/Drive Parameters -> Motor Characteristics.</td>
<td></td>
</tr>
</table>

<table>
<tr><td>9</td><td>Enter the motor data for the motor.</td><td>
</td></tr>
<tr><td>10</td><td>In the Communication menu, specify the CANopen address, which for this application will be a number between 8 and 13 and set the transmission speed to 500 kbit/s.</td><td>


</td></tr>
<tr><td>11</td><td>Now you can save the data.</td><td>
</td></tr>
</table>

12	Use either **Action->Download** **Or** **Right mouseclick on the folder name and select download** To download the configuration to the ATV31	
	Acknowledge the warning message with **OK**.	

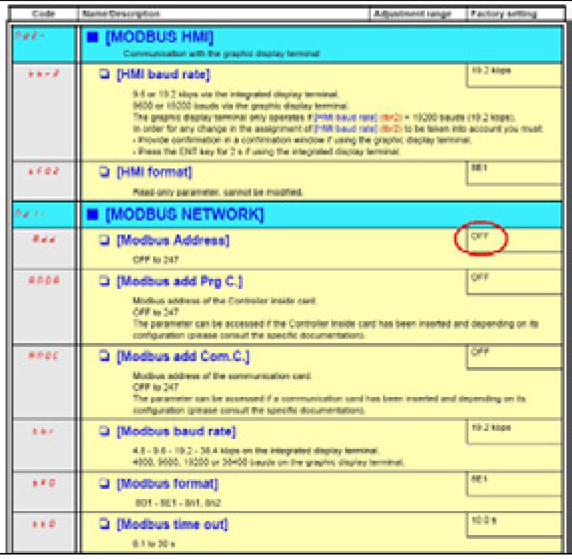
Altivar 71

Introduction

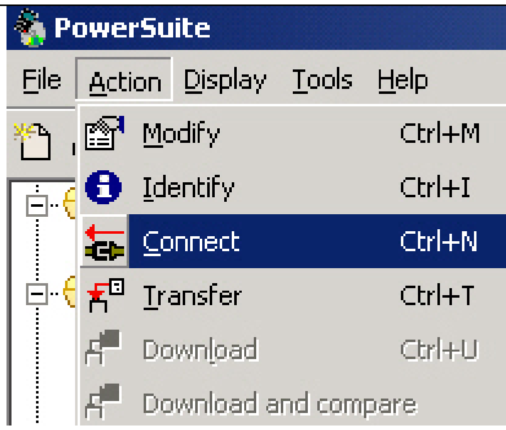
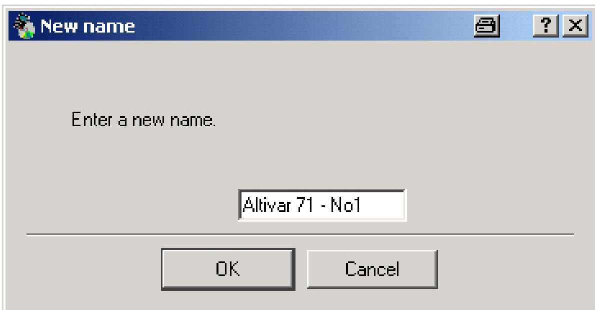
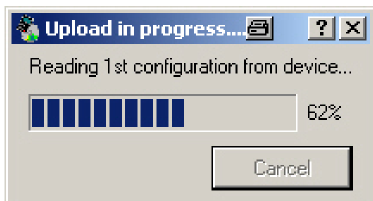
The settings for the ATV71 variable speed drives can either be made manually using the device's graphic display terminal or by means of the PowerSuite configuration software.

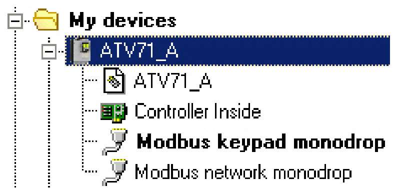
This section describes how to set the communication parameters manually, via the operator panel. You need to do this in order to enable parameter setting via the software. Then the parameterization option using the PowerSuite software is described.

ATV71 Manual Setup (Modbus)

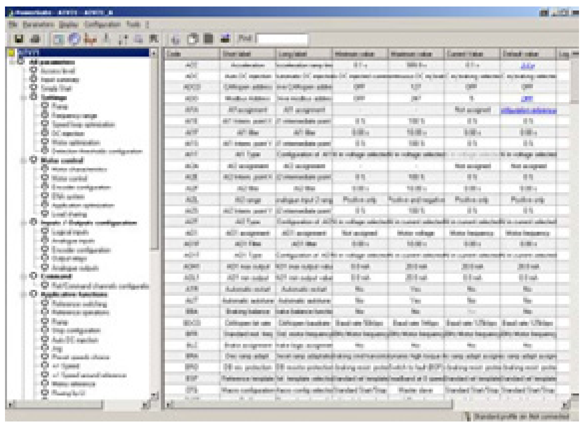
1	**ATV71** Installation Manual: (A PDF is supplied with the ATV71 on CD). The **Modbus addresses** on the interface are **factory-set** to **OFF**, i.e., the interface is inactive.	
2	From the main menu on the **ATV71 operator panel**, select: → **1.Drive Menu** → **1.9 Communication** → **Modbus Network** → **Modbus Address =** OFF → Press the selector switch to confirm. → Turn the selector switch to **MB-ADR = 1** and → Press the selector switch to confirm.	

PowerSuite with ATV71

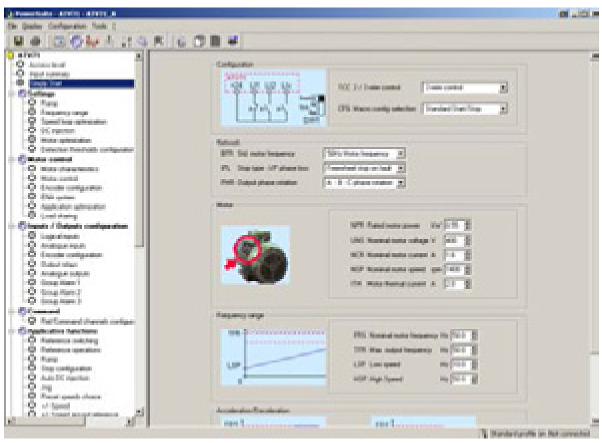
1	Select the **ControllerInside** folder in the project browser. Use**:** **Action->Connect** to connect to the ATV71. Make sure you have connected the ATV71 to your PC with the correct cable..	
2	Before the connection is established, you must confirm that you accept the terms of the security warning by pressing **ALT+F**.	
3	PowerSuite recognises that it is a new device and asks for permission to add it to the database. Click on **Create.**	
4	Next, you need to enter the name of the configuration or device.	
5	The data is read out from the ATV71.	

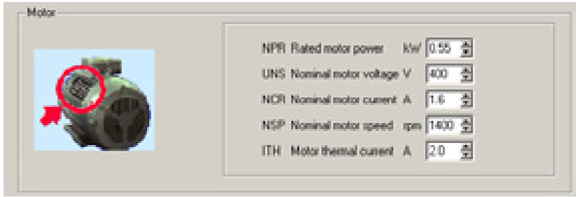
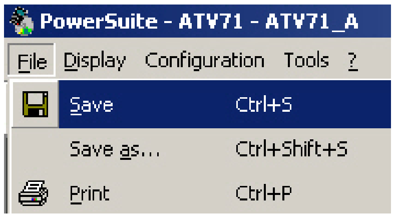
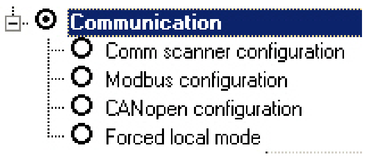
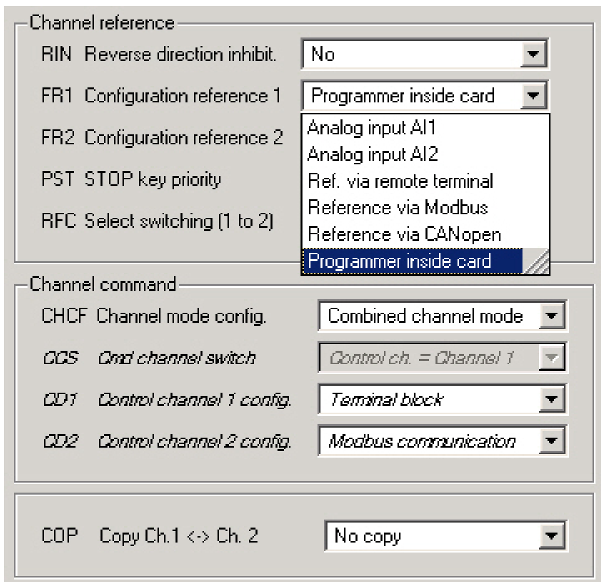
<table>
<tr><td>6</td><td>Once the transfer is complete, the device data will be displayed.</td><td>
ATV71_A
Characteristics
<table>
<tr><td>Reference</td><td>ATV71H075N4</td></tr>
<tr><td>Nominal power</td><td>0,75 kW</td></tr>
<tr><td>Supply Voltage</td><td>380 / 480 V</td></tr>
<tr><td>Maximum transient current</td><td>3,5 A</td></tr>
<tr><td>Continuous output current</td><td>2,3 A</td></tr>
</table>
</td></tr>
<tr><td>7</td><td>This screenshot shows the details of the ATV71 with the Controller Inside card.</td><td>
Structure
<table>
<tr><th>Card</th><th>Reference</th><th>Serial number</th><th>Version</th><th>Vendor name</th></tr>
<tr><td>Device</td><td>ATV71H075N4</td><td>921782131792565</td><td>V1.1IE01</td><td>TELEMECANIQUE</td></tr>
<tr><td>Control Board</td><td>Control part number</td><td>02461310248338</td><td>V1.1IE01</td><td>TELEMECANIQUE</td></tr>
<tr><td>Power Board</td><td>Power part number</td><td>2211280291</td><td>V1.1IE01</td><td>TELEMECANIQUE</td></tr>
<tr><td>Controller Inside</td><td>VW3A3501</td><td></td><td>V1.1IE03</td><td>TELEMECANIQUE</td></tr>
</table>
</td></tr>
<tr><td>8</td><td>You can select the relevant drive in the project browser on the left-hand side.</td><td>


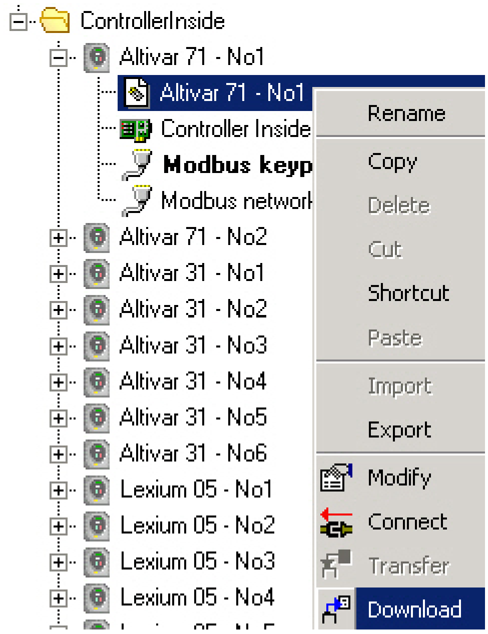
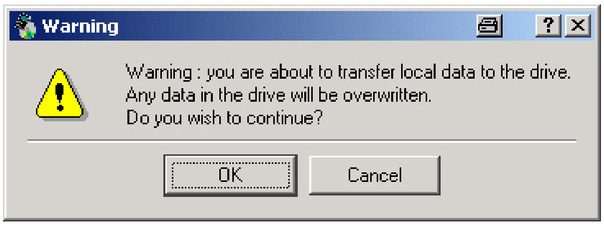
</td></tr>
<tr><td>9</td><td>The parameters can be displayed and modified in list format

OR….</td><td></td></tr>
<tr><td>10</td><td>….using graphical support.

The view shown here can be accessed via

Simply Start.</td><td></td></tr>
</table>

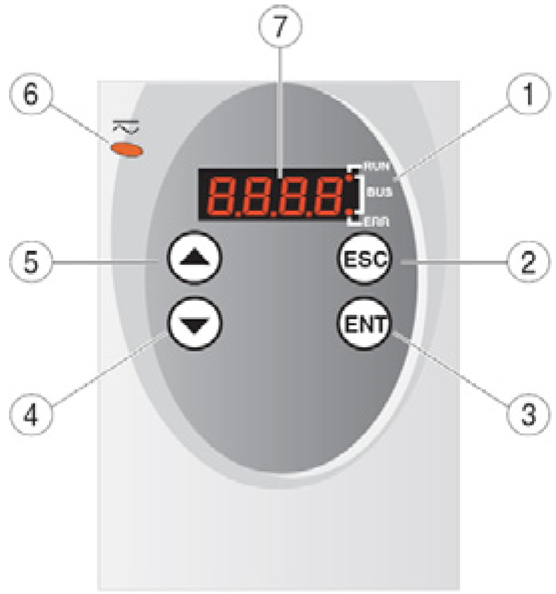
11	Enter the motor data for the motor.	
12	Now you can **save** the data.	
13	In the **Communication** area, the following distinction is made between the two ATV71 drives: **ATV71 with CI** 2x Modbus with Adr. 1 Adr. 8 (CI) No CANopen **ATV71 on CANopen** 1x Modbus 1x CANopen with address 2 and baud rate 500	
14	Under Command, you can define the command centre for the drives. ATV71 with CI: Programmer inside Card ATV71 on CANopen: Reference via CANopen	

15	Use: **Action->Download** OR Right mouse click on the Altivar 71 entry in the browser To download the configuration to the device Acknowledge the warning with **OK**.	 

Lexium 05

LXM05 Manual Setup

<table>
<tr>
<td>1</td>
<td>After wiring is complete, the drive control parameters must be set.

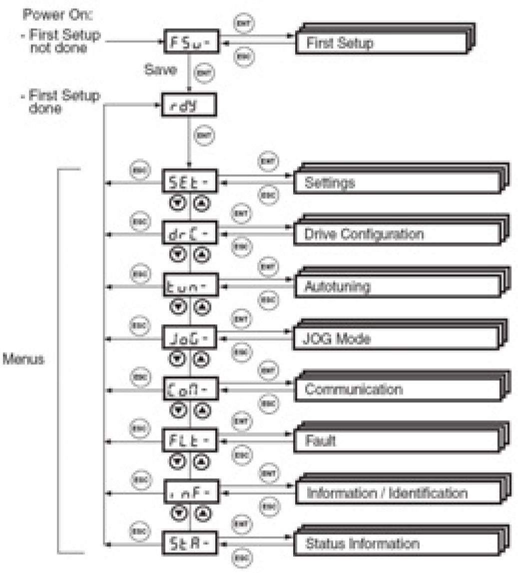
Parameters can be edited via the integral operating panel (HMI).</td>
<td>


(1) LEDs for fieldbus
(2) ESC:
- exit a menu or parameter
- return from the displayed to the last saved value
(3) ENT:
- call a menu or parameter
- save the displayed value to EEPROM
(4) Down arrow:
- switch to next menu or parameter
- reduce the displayed value
(5) Up arrow:
- switch to previous menu or parameter
- increase the displayed value
(6) Red LED on: DC bus under power
(7) Status display
</td>
</tr>
<tr>
<td>2</td>
<td>The HMI operates with menus.

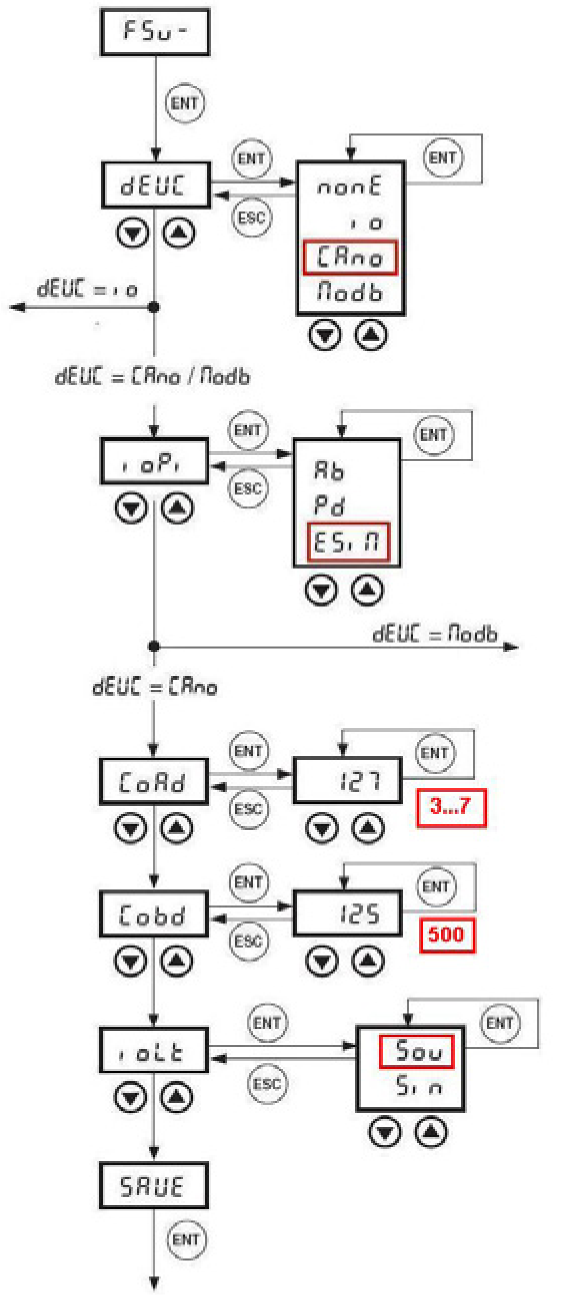
The screenshot to the right shows the top level of the menu structure.

In order to gain access via the PowerSuite software, you will first need to check the Modbus parameters.

Under CoM, make the following settings:

MbAd = 1
Mbbd = 19.2.</td>
<td>


</td>
</tr>
</table>

	When the drive is supplied with 24V for the first time, or if the factory settings have previously been loaded with the `PARfactorySet` parameter, all the drive functions are still blocked. **You must carry out an initial setup procedure.** To establish the link to the CANopen master, you will need to make settings in respect of the following: - **Method of control** - **Signal selection position interface** - **CANopen parameter and** - **Logic type** On completion the drive should always report "**`RDY`**" (ready) in the status display.	

PowerSuite mit LXM05

Besides using the control panel on the device, you can also use Powersuite to configure the Lexium05.

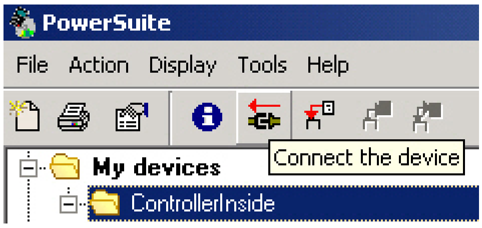
<table>
<tr><td>1</td><td>Use

Action->Connect

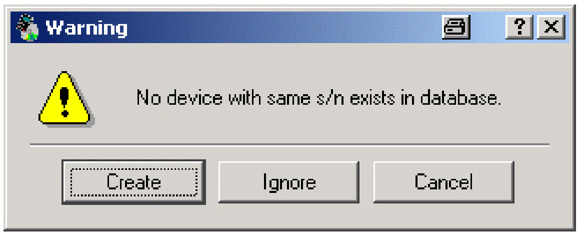
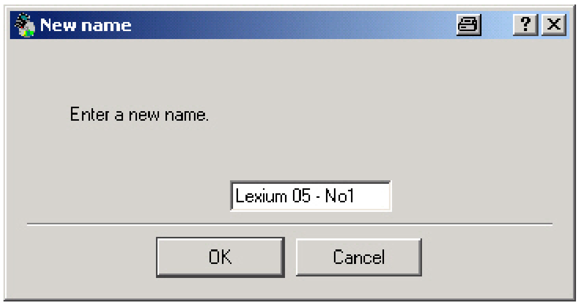
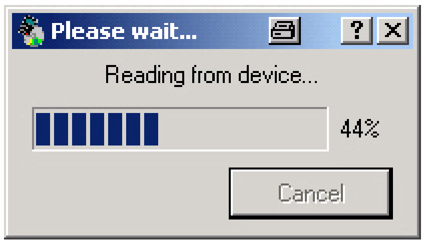
OR

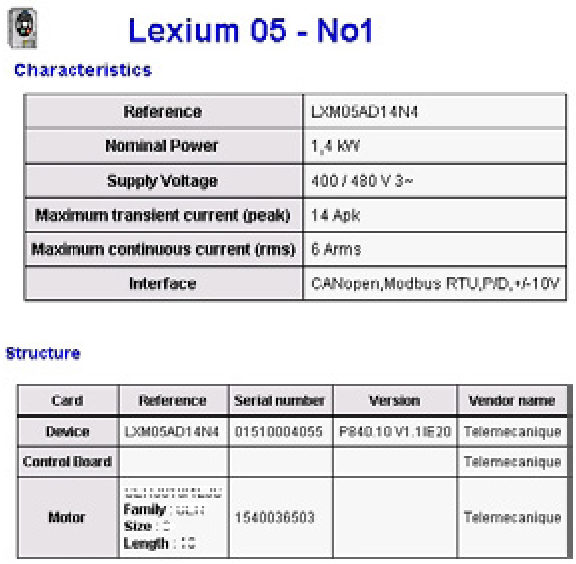
The icon in the toolbar

to connect to the Lexium05.

Remember to connect the PC to the lexium05 with the correct cable.</td><td>
</td></tr>
<tr><td>2</td><td>Once connected, PowerSuite recognises a new device and asks for confirmation to insert it into its database.

Acknowledge the request with

Create</td><td>
</td></tr>
<tr><td>3</td><td>Enter a new name for the device when requested and exit with OK.</td><td>
</td></tr>
<tr><td>5</td><td>The data is read out from the Lexium05.</td><td>
</td></tr>
</table>

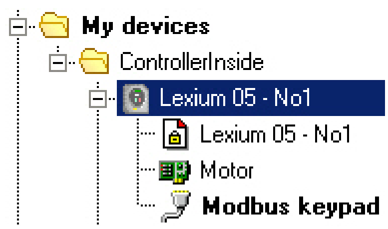
<table>
<tr>
<td>6</td>
<td>Once the transfer is complete, the device data will be displayed.</td>
<td>


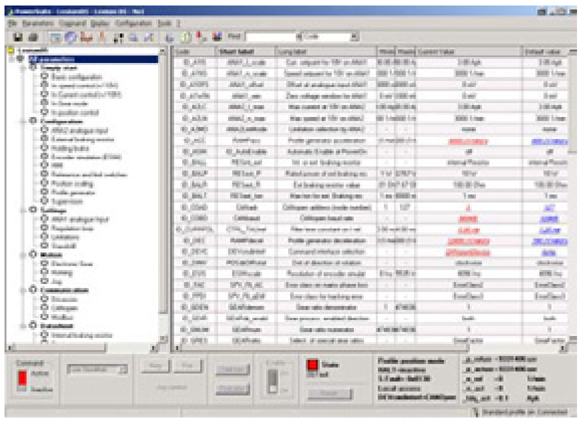
Lexium 05 - No1

Characteristics

Reference	LXM05AD14N4
Nominal Power	1,4 kW
Supply Voltage	400 / 480 V 3~
Maximum transient current (peak)	14 Apk
Maximum continuous current (rms)	6 Arms
Interface	CANopen,Modbus RTU,P/D,+/-10V

Structure

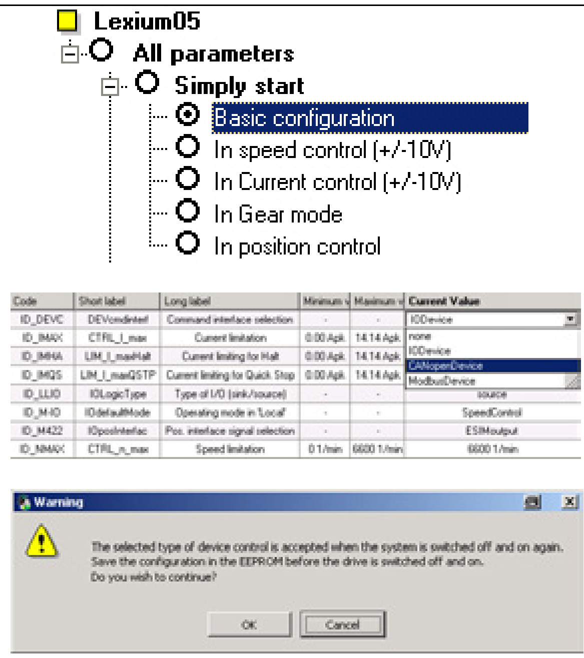
Card	Reference	Serial number	Version	Vendor name
Device	LXM05AD14N4	01510004055	P840.10 V1.1IE20	Telemecanique
Control Board				Telemecanique
Motor	Family : [illegible] Size : [illegible] Length : [illegible]	1540036503		Telemecanique
</td>
</tr>
<tr>
<td>7</td>
<td>You can select the relevant drive in the project browser on the left-hand side.</td>
<td>


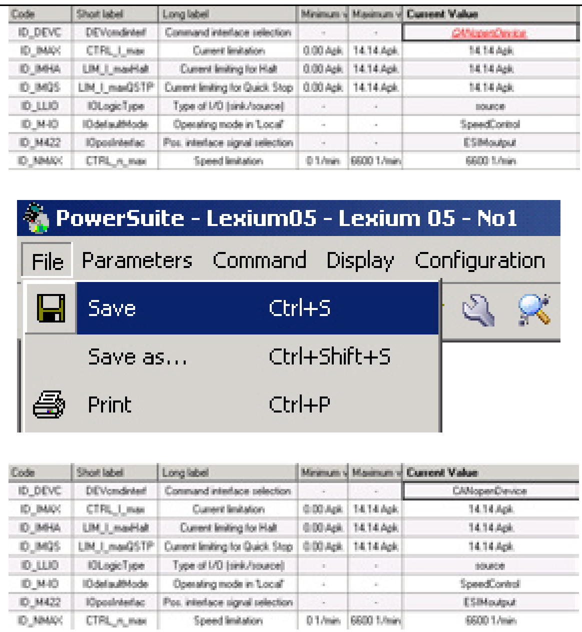
</td>
</tr>
<tr>
<td>8</td>
<td>The parameters can be displayed in list format or in page view.</td>
<td>

</td>
</tr>
</table>

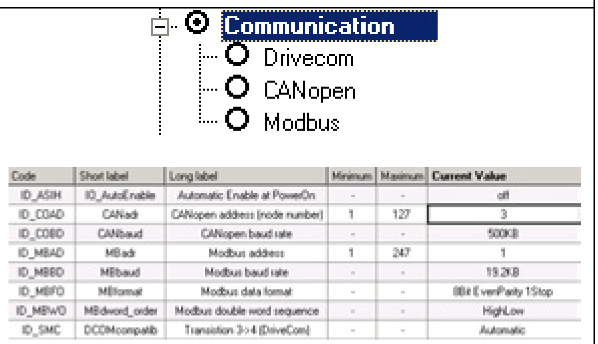
<table>
<tr>
<td>9</td>
<td>Select:
Simply start->Basic configuration.

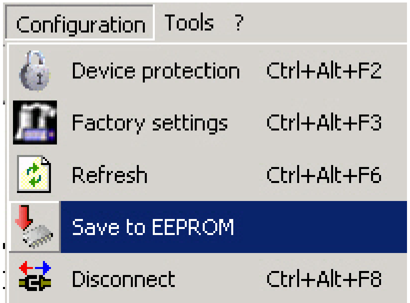
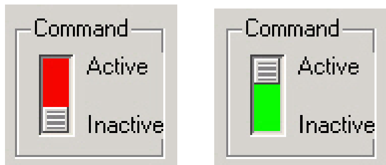
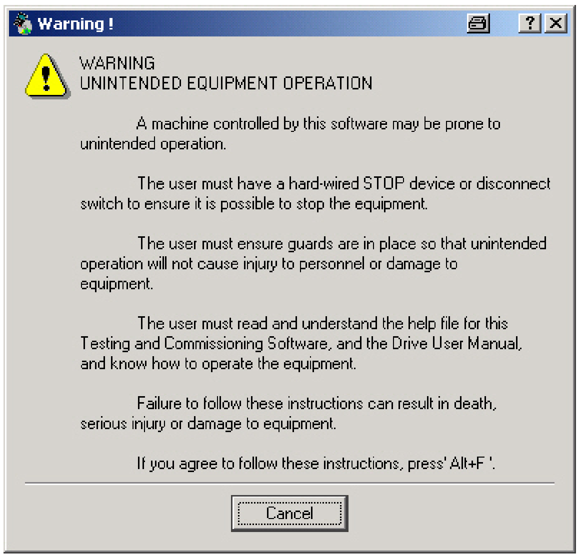
In the field for defining the control method for the device, you should select CANopenDevice.

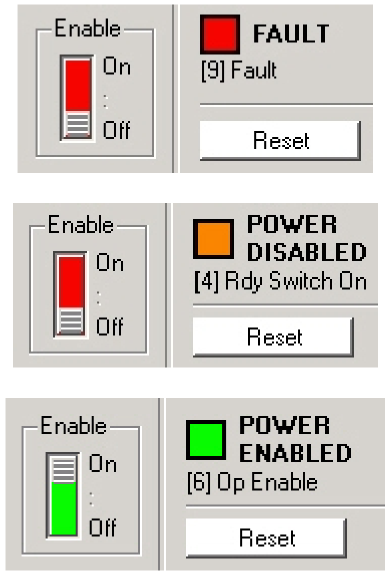
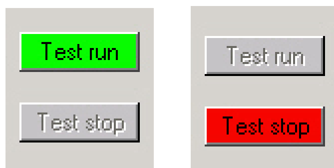
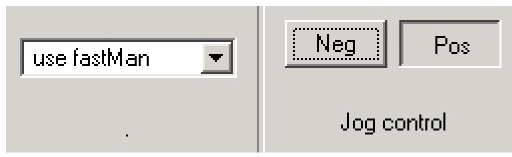
The servo drive will now be enabled for control via CANopen.

In order for this change to take effect on the Lexium05, you will need to switch the device off and then back on again.</td>
<td>
</td>
</tr>
<tr>
<td>10</td>
<td>Initially, the change will be highlighted in red, but the display color will change when you select:

File->Save.</td>
<td>
</td>
</tr>
<tr>
<td>11</td>
<td>Make the following settings under Communication:

CANopen address: 3...7
CANopen baud rate: 500
Modbus address: 1</td>
<td>
</td>
</tr>
</table>

12	To transfer the settings to the Lexium05, select: **Configuration -> Save to EEPROM.**	
13	The settings will now be transferred. To ensure that the settings are saved on the servo drive, you need to confirm the prompt by clicking **OK.** PowerSuite confirms the save when completed	 
14	You have the option of controlling the servo drive via the PowerSuite software. To be able to do this, you must first set the **Command** switch to **Active**. Press **Alt+F** to confirm the security warning.	 

15	Now set the **Enable** switch to **On**. Potential errors can be acknowledged by clicking **Reset.**	
16	**Test run** can be used to activate the servo drive. **Test stop** can be used to stop it again**.**	
17	Use the buttons **Neg** and **Pos** to change the direction of rotation on the drive.	
18	Information about the speed and position is displayed on the bottom right.	_p_refusr = 9893087 usr _p_actusr = 9890525 usr _n_ref = 180 1/min _n_act = 173 1/min _Idq_act = 0.07 Apk

Performance

Scan and Cycle time

A cycle time of 25 ms was not exceeded with the present configuration including the required application code.

The memory utilization of the PLC specified and used in this document was 40% for system data and 22% for the logic component.

Appendix

Detailed Component List

Hardware Components

Pos.	Amt.	Description	Part Number	Rev./ Vers.
1.1	1	3-pin master switch Compact NSC100 N	29003	
1.2	1	3-pin terminal cover	29321	
1.3	1	3-pin 12-16 A trip block, TM16D	29035	
1.4	1	230 V undervoltage release	29407	
1.5	1	Alarm accessories 1 NC contact	29450	
1.6	1	Fixed locking device	29371	
2.1	1	Preventa emergency off relay	XPSAF5130	
2.2	1	Emergency-off pushbutton housing	XALK178G	
2.3	7	Motor circuit breaker (9 - 14 A) for LEX05 and ATV71	GV2ME16	
2.4	6	Motor circuit breaker (2.5 A) for ATV31	GV2L07	
2.5	2	Motor contactor	LC1D18BD	
2.6	1	Auxiliary switch block	LADN11	
2.7	3	24 V DC, 5 A power supply	ABL7RE2405	
2.8	3	2-pin, 2 A back-up fuse for 24 V DC	GB2DB07	
2.9	1	1-button pushbutton housing, empty	XALD01	
2.10	1	Illuminated button blue, flat	ZB5AW363	
2.11	1	Auxiliary switch block with lamp holder plus white LED	ZALVB1	
2.12	1	Standard auxiliary switch (1xNO)	ZBE101	
2.13	1	Standard auxiliary switch (1xNC)	ZBE102	
3.1	2	Altivar 71 variable speed drive	ATV71H075N4	**V1.1**
3.2	1	Controller Inside card for ATV71	VW3A3501	
3.3	6	Altivar 31 variable speed drive	ATV31H037N4	**V1.2**
3.4	5	Lexium05 servo drive	LXM05AD14N4	**V1.1**
3.5	5	Servo motor	BSH0702P31A2A	
3.6	5	Servo motor power cable	VW3M5101R30	
3.7	5	Servo motor feedback cable	VW3M8101R30	
3.8	1	Magelis operator and display terminal	XBTGT2330	
4.1	1	CANopen bus adapter	STBNCO2212	
4.2	**0**	Bus terminating resistor included with bus adapter	STBXMP1100	
4.3	1	Field power supply	STBPDT3100	
4.4	2	Digital input card, 6 inputs	STBDDI3610	
4.5	1	Digital input card, 4 inputs	STBDDI3420	
4.6	2	Digital output card, 6 outputs	STBDDO3600	
4.7	1	Module base for field power supply	STBXBA2200	
4.8	5	Module base for I/Os, type1	STBXBA1000	
4.9	1	24 V power connector, NIM (pack size = 10 units)	STBXTS1120	
4.10	1	24 V power connector, PDM (pack size = 10 units)	STBXTS1130	
4.11	1	I/O connector set (pack size = 20 units)	STBXTS1100	

Hardware Components Contd.

Pos.	amt.	Description	Part Number	Rev./ Vers.
5.1	1	PC – HMI programming cable	XBTZG935	
5.2	1	Modbus Hub	LU9GC3	
5.3	2	Modbus Cable (3,0 m) 2xRJ45	VW3A8306R30	
5.4	1	Modbus T-junction with cable (0,3m)	VW3A8306TF03	
5.5	2	Terminal resistor	VW3A8306RC	
5.6	**0**	PC - Advantys STB programming cable (supplied with software)	STBXCA4002	
5.7	1	Connection cable set for Altivar31, 71 and Lexium05	VW3A8106	
6.1	6	CANopen junction box	VW3CANTAP2	
6.2	1	CANopen cable (50m); also available in other versions and lengths	TSXCANCA50	
6.3	12	Pre-assembled CANopen cable (1 m) with RJ45 connector for ATV/LXM	VW3CANCARR1	
6.4	1	CANopen connector for ATV71/CANopen master	TSXCAN KCDF180T	
6.5	1	CANopen connector for Advantys STB	TSXCAN KCDF90TP	

Software Components

Pos.	Amt.	Description	Part Number	Rev./ Vers.
1.1	1	PS1131 PLC programming software with CoDeSys	See note [1)]	V2.30
1.2	1	Vijeo Designer for HMI	VJDSSDTGSV4 3M	V4.30
1.3	1	Advantys Configuration Software incl. RS232 connection cable	STBSPU1000	V2.00
1.4	1	PowerSuite parameterization software	VW3A8104	V2.30
1.5	1	PowerSuite LXM05 Launch parameterization software	Launch version supplied with LXM05	V2.20
1.6	1	PowerSuite ATV71 Launch parameterization software	Launch version supplied with ATV71	V2.20

Note:

1) The part number will be announced at the special training course focusing on the programmable Controller Inside card.

Component Protection Classes

Mounting Location / Protection Class

Component	In the Field IP55/IP65	Frontside IP65	Cabinet IP20
Master switch with or without undervoltage release fuse and built-in trip indicator			X
Emergency-off switch housing (XALK)	X		
Preventa modules (XPSxx)			X
Motor protection switch, all types and ratings			X
Contactors, LC1			X
XALD pushbutton housing, with components fitted	X		
Phaseo power supplies 24 V DC			X
Altivar 71 variable speed drives, all rating classes			X
Altivar 31 variable speed drives, all rating classes			X
Lexium05 servo drive, all rating classes			X
Servo motor	X		
Magelis XBTG graphic panel, all versions		X	
Advantys STB distributed I/O island			X
CANopen TAPS with CAN cable			X

Component Features

Components

Preventa safety relay: XPSAF5130



- Category 3 to EN 954 Part 1
- 24 V DC
- 3 safety-oriented switching contacts
- 1 semiconductor output for PLC
- Slimline design

Phaseo power supply unit: ABL7RE2405



- 100 to 240 V AC/24 V DC
- 5 A secondary, other ratings also possible
- Slimline design
- Parallel connection possible
- Short-circuit-proof and protected against overload

Altivar variable speed drive: ATV71H075N4



- 0.75 kW, 400 V AC three-phase
- Integrated class B EMC filter
- Temperature range: -10 to +50°C
- Speed range 0 to 1000 Hz
- Graphic display for control and parameterization
- Operation via Modbus, CANopen or other buses possible
- Option cards: communication, encoder, Controller Inside
- 2 analog inputs plus 1 analog output
- Digital inputs, 2 digital status outputs
- 1 shutdown output (emergency-off function)
- Expansion cards for buses, I/O, control
- Protection of drive and motor
- Compact design, side-by-side installation possible

Programmable Controller Inside card VW3A3501



- PC interface for programming with PS 1131 dialog tools
- 24 V DC power supply
- 1 CANopen bus master interface
- 10 logic inputs, 2 of which can be used for 2 counters or 4 of which can be used for 2 incremental encoders
- 2 analog inputs
- 6 logic outputs
- 2 analog outputs
- 5 signaling LEDs

Components Contd.

Altivar variable speed drive: ATV31H037N4



- 0.37 kW, 380 to 500 V AC three-phase
- Integrated class B EMC filter
- Temperature range: -10 to +50°C
- Speed range from 1 to 20 (0 to 200 Hz)
- Speed control with flow vector check
- Operation via Modbus or CANopen possible
- 2 analog inputs plus 1 analog output
- Digital inputs
- 2 or 3 digital status outputs possible
- Protection of drive and motor
- Compact design, side-by-side installation also possible on a DIN rail using bracket VW3A11852

Lexium05 servo drive: LXM05AD14N4



- 1.4 kW, 380 to 480 V AC three-phase
- Integrated EMC line filter
- Temperature range: -10 to +50°C
- Operation via CANopen or Modbus possible
- 4 operating modes
  - Point-to-point (relative or absolute)
  - Speed or torque control
  - Electronic gears
  - Manual mode
- 2 analog inputs +/- 10 V
- 4 digital inputs and 2 digital outputs
- Protection of drive and motor

Servo motors: BSH0702P31A2A



- Continuous/peak/rated torque: 2.12/5.63/1.6 Nm
- Rated speed: 6000 rpm
- BSH servo motors can be supplied in the following versions:
  - IP40 or IP65 degree of protection
  - With or without holding brake
  - Straight or right-angled connector
  - Singleturn or multiturn SinCos encoder
  - Smooth shaft or shaft with featherkey
- Degree of protection
  - Motor enclosure: IP65 in accordance with IEC/EN 60529
  - Shaft end: IP40 or IP65 in accordance with IEC/EN 60529
- Integrated sensor, Hiperface® SinCos absolute encoder (singleturn or multiturn) with high-resolution interface
- Smooth or stepped shaft end, standard size (according to DIN 42948)

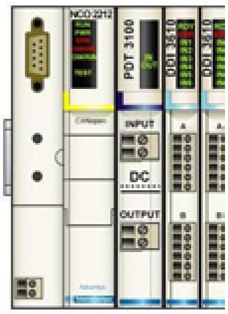
Components Contd.

Operator and display terminal: Magelis XBT GT2330



- 5.7" color touch panel (65536)
- 24 V DC power supply (screw terminals)
- 9-pin Sub-D socket (COM 1) for serial link via RS232C or RS485
- RJ45 socket (COM 2) for serial link
- RJ45 connector for Ethernet link
- USB Port
- Slot with cover for compact flash memory card
- Interface for expansion unit (for future use)

Advantys STB



- Modular I/O system with
  - Various fieldbus couplers
  - Analog and digital modules
  - Counters
  - Expert modules
- I/O selectivity 2 to 16 channels
- Open to motor starters, variable speed drives and devices thanks to the bus backplane
- Product dimensions
  - General 120 x 78 mm (HxD)
  - Fieldbus coupler 40.5 mm wide
  - I/Os and power module 13.5 or 18 mm wide
- Advantys STB has 7 different fieldbus interfaces, also called NIM
  - CANopen
  - Ethernet TCP/IP
  - DeviceNet
  - Modbus Plus
  - INTERBUS
  - FIPIO
  - Profibus DP
- Other features
  - Removable memory card
  - I/O modules modified and parameterized via the serial interface
  - Local HMI can be connected via the serial interface
  - Direct, on site open-loop and closed-loop control, even if communication with the PLC fails
  - HotSwap
  - Separate infeed for input and output voltage
  - Detailed diagnostics
  - Reflex functions (intelligent preprocessing)

Components Contd.

PLC programming software with CoDeSys: PS1131



The PS1131 dialog tool conforms to international standard IEC 61131-3. It contains all the functions necessary for programming and setting up the programmable Controller Inside card.

- It also contains the CANopen Configurator.
- The programming and testing tools can be accessed via the application navigator. It provides an overview of the entire program and enables rapid access to all components of the application:
    - Program editor
    - Function block editor
    - Variables editor
    - Animation table editor
    - Operating screen editor

- Software supports the mono-task structure (cyclic or periodic) It is made up of several subprograms
- Data is exchanged with the variable speed drive via a function block. This block is included in the standard library.
- The following 6 programming languages are available:
  - Ladder Language (LD)
  - Structured Text (ST)
  - Grafcet (SFC)
  - Instruction List (IL)
  - Function Block Diagram (FBD)
  - Continuous Function Chart (CFC)
- The key testing functions are:
  - Use of breakpoints
  - Step-by-step program execution
  - Individual cycle execution
  - Direct access to called subprograms (call register)
- Application testing
  - Program execution (breakpoints, step-by-step program execution, etc.)
  - Animation tables
  - Oscilloscope
  - Operating screens (dedicated operating screens with animation of graphic objects, which are assigned to variables)
  - Simulation (with drive disconnected)

Components Contd.

Advantys Configuration Software: STB SPU 1000



Proceed as follows to configure an Advantys STB system:

- If applicable, parameterize all the I/O modules on the Advantys STB platform (digital, analog and intelligent modules) with standard functions.
- Generate the dedicated island EDS file, which will be used at a later stage
- Parameterize the reflex functions executed at island level. These parameters are set using the Advantys Configuration Software.

This software can also be used:

- To optimize island performance by specifying priorities to be applied when processing module data
- To add preferred modules or standard CANopen devices (such as FTB, OTB, ATV31, Lexium05, for example)
- To check that the configuration complies with the design guidelines and to check the current consumption
- To change the module's standard functions.

Vijeo Designer



Vijeo Designer configuration software has a number of parameterization windows that enable a project to be developed quickly and simply and are very user-friendly. Vijeo Designer uses Java scripts that allow process data to be further processed on the XBT-G touch panel.

These are some of its functions:

- Navigator
- Library of animated graphic objects
- Online help
- Display of error reports
- Display of object characteristics
- Display of the list of variables

Components Contd.

PowerSuite dialog tool



- The PowerSuite dialog tool enables user-friendly operation of the following devices:
  - Altivar variable speed drive
  - Lexium05 servo drive
  - TeSys model U motor controls
  - Altistart soft starter
- A wide range of functions are integrated for the various application phases, e.g.:
  - Preparing the configurations (for PC),
  - Commissioning (for PC and Pocket PC)
  - Maintenance (for PC and Pocket PC)
- During the startup phase, the device is connected to a PC and can be used:
  - To transfer the configuration that has been made
  - To make settings
  - For monitoring. New functions have now been added for this option such as the Oscilloscope function.
  - For control
  - To save the final configuration

Contact

Author	Telephone	E-Mail
Schneider Electric GmbH Customer & Market System & Architecture Architecture Definition Support	+49 6182 81 2555	cm.systems@de.schneider-electric.com

Schneider Electric GmbH
Steinheimer Strasse 117
D - 63500 Seligenstadt
Germany

As standards, specifications and designs change from time to time, please ask for confirmation of the information given in this publication.